X-641-73-202
PREPRINT

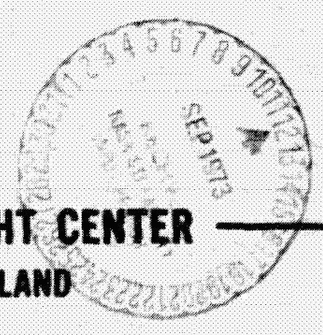
NASA TM X- 70438

THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS BY THE TAYLOR SERIES METHOD

ALLAN SILVER
EDWARD SULLIVAN

JULY 1973

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND



(NASA-TM-X-70438) THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS BY THE TAYLOR SERIES METHOD (NASA) 76 p HC $6.00
CSCL 12A
N73-30567
Unclas
G3/19 12450

The Numerical Solution of Ordinary Differential Equations by the Taylor Series Method

Allan Silver
and
Edward Sullivan

Laboratory for Space Physics
NASA-Goddard Space Flight Center
Greenbelt, Maryland 20771

I. Introduction

The Taylor series method [1] has long been regarded as an efficient procedure for solving systems of ordinary differential equations. Frequently, it is necessary to algebraically manipulate the differential system into an equivalent system. The Taylor coefficients for this modified system may be simply written. However, the required modification is a tedious and error prone task for all but the simplest systems. For this reason, the Taylor series method has often been excluded by numerical analysts from consideration as a general purpose integrator.

In Moore [2], the procedures for recasting a system, which is reducible to a rational form, have been described in detail. Barton, Willers, and Zahar [3] describe techniques for automatic step length prediction, local error estimation, and for choosing the proper number of terms in the series. The authors also include a comparison of the Taylor series method with the fourth order Runge-Kutta method [4] and the Bulirsch-Stoer rational extrapolation method [5]. For a wide range of accuracy, it was found that the Bulirsch-Stoer method generally required five times the amount of computing time, and the factor for the Runge-Kutta method varied from five to one hundred.

A method for the automatic reduction of arbitrary

differential systems is described in Barton, Willers, and Zahar [6]. Also presented is a procedure to generate the computer routine which evaluates the Taylor series coefficients of the reduced system. The system reduction and program generation are analogous to the output from a compiler, and the differential equations and initial values are treated as simple language statements that are input to the "compiler". The particular implementation in [6] is an interactive program written for the Atlas 2 computer in Cambridge, England. The target language is Atlas machine language code.

In this paper, an implementation allowing wider usage is presented. The "compiler" is written in PL/I, and the target language is Fortran IV. In Section II, the reduction of a differential system to rational form is described along with the procedures required for automatic numerical integration. In Section III, the Taylor series method is compared with the Bulirsch-Stoer method and with the Nordsieck version of the Adams predictor-corrector method [7] for a number of differential equations.

In Section IV, algorithms using the Taylor series method to find the zeroes of a given differential equation and to evaluate partial derivatives are presented.

Section V discusses the PL/I implementation of the

Barton et al. algorithm. Appendix A contains an annotated listing of the PL/I program which performs the reduction and code generation. Included in Appendix B are listings of the Fortran routines used by the Taylor series method. Finally, Appendix C has a compilation of all the recurrence formulas used to generate the Taylor coefficients for non-rational functions which may appear in the defining system of equations.

II. The Taylor Series Method

Consider the following differential system

$$\frac{d\bar{y}}{dt} = \bar{f}(t,\bar{y}),\ \bar{y}(a)=\bar{\alpha},\ a\leq t\leq b \tag{2.1}$$

where the f_i are rational functions. To apply the Taylor series method to this system, the Taylor coefficients for the expansion about the point $t_o=a$ are computed. The dependent variables y_i are then evaluated at $t=t_1$, with

$$y_i(t_1)=\sum_{j=o}^{\infty} \frac{d^j y_i(t_o)}{dt^j}\ \frac{(t_1-t_o)^j}{j!} \tag{2.2}$$

The value t_o is now replaced by t_1 and the process repeated until the y_i at the value $t_1=b$ are evaluated

Initially, it may appear that the applicability of the method only to differential systems involving rational functions is a severe limitation on the usefulness of the method. However, functions such as sin, cos, exp, etc., are solutions of rational differential systems. Consequently, a large class of solutions of non-rational differential systems have equivalent representations as solutions of rational differential systems.

To illustrate this point, the function y satisfying the differential equation

$$\frac{dy}{dt} = e^{\sin(y)+e^{\cos(y)}},\ y(0)=0,\ 0 \leq t \leq \frac{\pi}{2} \tag{2.3}$$

may be written as the function u_1 in the system

$$\begin{aligned}
\frac{du_1}{dt} &= u_4+u_5 \\
\frac{du_2}{dt} &= u_3(u_4+u_5) \\
\frac{du_3}{dt} &= -u_2(u_4+u_5) \\
\frac{du_4}{dt} &= u_4u_3(u_4+u_5) \\
\frac{du_5}{dt} &= -u_5u_2(u_4+u_5) \\
u^f(0) &= [0,0,1,1,e] \quad , \qquad 0 \le t \le \frac{\pi}{2}
\end{aligned} \tag{2.4}$$

where $u_2=\sin(u_1)$, $u_3=\cos(u_1)$, $u_4=e^{u_2}$, $u_5=e^{u_3}$.

To obtain the canonical system equivalent to (2.4), auxiliary variables are introduced so that each equation in the canonical system represents a single operation of either addition, subtraction, multiplication, or division. Once the canonical system has been generated and the order of evaluation determined, it is a simple task for the computer to produce the formulas for the coefficients.

In the implementation of the method it is important to determine how to best evaluate expressions of the form

$$y_i(t_1) = \sum_{j=o}^{j_{max}} y_i^{(j)}(t_o)(t_1-t_o)^j \tag{2.5}$$

where $y_i^{(j)}(t_o) = \frac{1}{j!}\frac{d^j y_i}{dt^j}(t_o)$.

Also, it is necessary to decide whether j_{max} should be a constant value over the interval of integration or whether j_{max} should be changed at each integration step. Other questions involve the procedure for varying step length and the method of estimating local truncation error.

It was found for a number of test differential equations, including those in Section III, that Horner's method [8] for evaluating (2.5) proved to be the most efficient. Horner's method applied to (2.5) is given by

$$\begin{aligned} &\text{a)} \quad y_i(t_1) = y_i^{(j_{max})}(t_o)\ (t_1-t_o) \\ &\text{b)} \quad y_i(t_1) = y_i^{(j)}(t_o) + (t_1-t_o)\,y_i(t_1) \\ &\qquad \text{for } j = j_{max}-1, \ldots, 0 \end{aligned} \tag{2.6}$$

Relative error in the Taylor series solution is controlled by methods analogous to those commonly used for other discrete integrators. The interval length is varied from step to step in order to yield a local relative truncation error less than some preset error bound. The error term resulting from the truncation to j_{max} terms of the Taylor series for $y_i(t_1)$ expanded about t_o is

$$y_i^{(j_{max}+1)}(\xi)\,(t_1-t_o)^{j_{max}+1} \qquad t_o \leq \xi \leq t_1$$

Thus, a local relative error bound of E requires that the step length $h = t_1 - t_o$ satisfy

$$h^{j_{max}+1} \leq E \min_i \left[\left| \frac{N_i}{y_i^{(j_{max}+1)}(t_o)} \right| : N_i = \begin{cases} y_i(t_o) & \text{for } y_i(t_o) = 0 \\ 1 & \text{for } y_i(t_o) = 0 \end{cases} \right] \tag{2.7}$$

where i varies over the set of indices for which $y_i^{(j_{max}+1)}(t_o) \neq 0$. If $y_i^{(j_{max}+1)}(t_o) = 0$ for all i, then h is set to step to the end of the range.

For the differential equations considered in Section III, the fixed j_{max} which proved to be most efficient was equal to the number of significant decimal digits carried by the computer. This was also found to be true for the equations tested in [6]. For many problems where large functional changes occur over the integration interval, and computation time is critical, a variable j_{max} may produce a very efficient procedure. For a further discussion of numerical integration methods which are optimized by changing the order at each step, see [9] and [10].

III. Comparison

An age old problem confronting numerical analysts is the generation of effective procedures for the comparison of computational methods. It is virtually impossible to include such characteristics as simplicity of method, implementation effort, reliability, and efficiency in a conclusive evaluation. Almost all comparisons of numerical integrators are made solely on the basis of efficiency - usually measured by the number of integration steps or the computer time required to obtain solutions of equal accuracy.

With third generation machines, the concurrent execution of programs, and optimizing compilers, the computer time required for solution is subject to wide fluctuations. These fluctuations are often of the same order of magnitude as the computation times being measured. Also, during the computation, there is an overhead charge incurred when index registers are saved, arguments are passed, and loops are generated.

Many implementations of a numerical algorithm will reduce the overhead at the expense of generality. It is unfair to compare on the basis of computer time, routines which differ in their implementation philosophy, because for the moderate sized problems generally used as test cases the overhead is often a significant portion of the computation time. Consequently, a less general, low overhead method may perform

competitively with a less efficiently programmed and more general method.

In comparing the Taylor series method with other methods, significant factors such as the extra storage needed, the difficulty in learning to use the "compiler", and the effort in debugging the Taylor coefficient routine if the "compiler" malfunctions, are difficult to include. Further difficulties result because the Taylor series method integrates a different system of equations than do the usual methods.

To eliminate implementation dependence from the estimate of a method's efficiency, each test problem was integrated to determine the number of derivative evaluations required for solution. This number should be approximately constant for a given method regardless of implementation. For each derivative evaluation routine, the number of machine (360/91) cycles required for one pass through this routine was determined. Table III-A shows the number of cycles required for some typical operations.

TABLE III-A

OPERATION	NUMBER OF CYCLES*
D.P. Load and Store	0
D.P. Add and Subtract	2
D.P. Multiply	3
D.P. Divide	12
F.P. Add, Subtract, Load and Store	1
F.P. Multiply	11
F.P. Divide	37
D.P. Sin and Cos	217
D.P. Exponential	217
D.P. Square Root	133
D.P. Power	400

F.P. = Integer Arithmetic
D.P. = Double Precision Floating Point Arithmetic

*Not including overlap or simultaneous operations.

The number of cycles required to pass arguments from the calling routine was not counted and the overlap or simultaneous execution of operations was not considered.

The methods selected for comparison were the Bulirsch-Stoer rational extrapolation and the Nordsieck version of the Adams predictor-corrector. Both of these methods require a number of functional evaluations to obtain a starting step size which satisfies the accuracy condition. If the initial estimate for the step size is far off, the number of evaluations used in starting could be quite large. For the problems considered here, the number of evaluations required to start were not counted. The test problems are five representative non-trivial differential equations encountered in a computation laboratory:

Problem 1. Bessel Function

$$Y'' = Y(2/t^2 - 1)$$
$$Y(0) = 0$$
$$Y'(0) = 0$$
$$Y''(0) = 2/3$$
$$0 \leq t \leq 25\pi/4$$

Solution: $Y(t) = \sin(t)/t - \cos(t) = tj_0(t)$

Problem 2. Coulomb Function [11]

$$Y'' = (-1 + 1/t)Y$$
$$Y(0) = 0$$
$$Y'(0) = (\Pi/(e^{\pi}-1))^{1/2}$$
$$0 \leq t \leq 20$$

Solution: $Y(t) = F_0(1/2,t)$

Problem 3. Restricted 3-body problem [5]

$$X'' = X + 2Y' - a'(X + g) - a(X - g')$$
$$Y'' = Y - 2X' - a'Y - aY$$
$$X(0) = 1.2$$
$$X'(0) = 0$$
$$Y(0) = 0$$
$$Y'(0) = -1.04935750983$$
$$g = 1/82.45,\ g' = 1 - g$$
$$a = g/((X - g')^2 + Y^2)^{3/2},\ a' = g'/((X + g)^2 + Y^2)^{3/2}$$
$$0 \leq t \leq 6.192169331396$$

Solution: The given range for t is one period.

Problem 4.

$$Y' = -Y + (1 + t)\cos(te^t)$$
$$Y(0) = 0$$
$$0 \leq t \leq 5$$

Solution: $Y(t) = e^{-t}\sin(te^t)$

Problem 5. A stiff equation [12]

$$X' = -2000X + 1000Y + 1000$$
$$Y' = X - Y$$
$$X(0) = 0$$
$$Y(0) = 0$$
$$1 \leq t \leq 4$$

Solution: $X(t)=1+A_1e^{-\lambda_1 t}+A_2e^{-\lambda_2 t}$

$Y(t)=1+B_1e^{-\lambda_1 t}+B_2e^{-\lambda_2 t}$

$\lambda_1=+2000.5001 \ldots$

$\lambda_2=+.49987500 \ldots$

$A_1=-.49975000 \ldots$

$A_2=-.50024999 \ldots$

$B_1=+.00024993746 \ldots$

$B_2=-1.0002499 \ldots$

Table III-B lists some of the results of testing four of the five problems. The column labeled "error" refers to the relative error of the computed solution at the end of the interval. "Cycles" is the number of machine cycles required for each evaluation. "DE" refers to the number of evaluations required to integrate over the given interval. Finally, the column labeled "R" contains the ratio

$$\frac{DE\left(\begin{smallmatrix}\text{Comparison}\\ \text{Method}\end{smallmatrix}\right) \times \text{Cycles}\left(\begin{smallmatrix}\text{Comparison}\\ \text{Method}\end{smallmatrix}\right)}{DE\left(\begin{smallmatrix}\text{Taylor Series}\\ \text{Method}\end{smallmatrix}\right) \times \text{Cycles}\left(\begin{smallmatrix}\text{Taylor Series}\\ \text{Method}\end{smallmatrix}\right)}$$

TABLE III-B

TAYLOR SERIES METHOD

PROBLEM	ERROR	CYCLES	DE
1	1.2×10^{-10}	789	15
2	4.1×10^{-9}	736	10
3	2.7×10^{-9}	23769	103
4	4.7×10^{-9}	2524	557

NORDSIECK METHOD

PROBLEM	ERROR	CYCLES	DE	R
1	7.9×10^{-10}	22	661	1.2
2	5.5×10^{-9}	19	741	1.9
3	1.0×10^{-9}	349	2340	0.1
4	2.4×10^{-9}	445	18374	5.8

BULIRSCH-STOER RATIONAL EXTRAPOLATION METHOD

PROBLEM	ERROR	CYCLES	DE	R
1	1.7×10^{-10}	22	1288	2.4
2	1.6×10^{-9}	19	790	2.0
3	1.0×10^{-9}	349	5769	0.8
4	1.6×10^{-9}	445	6612	2.1

where the comparison method is either Nordsieck or Bulirsch-Stoer.

The Taylor series method is superior to Nordsieck and Bulirsch-Stoer on Problems 1, 2, and 4 and inferior on Problem 3. Other results, not presented here, show that the Nordsieck and Bulirsch-Stoer methods are very inefficient for Problem 5, while the Taylor series method handles this problem well.

Once the user masters the fairly simple art of setting up the input for program generation, he has an easy means for applying the Taylor series method. If greater efficiency is required, the program may be optimized by the user who has some knowledge of Fortran. On the other hand, if an error occurs, the program may be difficult to debug. Finally it should be noted that there exists an important class of problems where no Taylor series method program can at present be generated. In general, however, the method is a valuable tool for solving many problems and is certainly worth trying.

IV. Finding Zeros; Partial Differentiation

In this section two algorithms are presented which may be used in conjunction with the Taylor series method. The first algorithm finds the zeros of a function and the second algorithm is used to find the partial derivatives of a function of several variables.

The method used to solve for the zeros of a function is that of series inversion. The relevant theorem is quoted here without proof [13].

Given the power series

$$f=f_o+\sum_{k=1}^{\infty} a_k(t-t_o)^k \tag{4.1}$$

with positive radius of convergence and $a_1 \neq 0$, then there exists a unique power series

$$t=t_o+\sum_{k=1}^{\infty} b_k(f-f_o)^k \tag{4.2}$$

with positive radius of convergence and such that the two series are inverses in sufficiently small neighborhoods of t_o and f_o and $b_1 = 1/a_1$.

To develop a recursion formula for the coefficients b_k in (4.2), solve for $(f-f_o)$ in (4.1) and substitute into (4.2), resulting in

$$t-t_o=\sum_{k=1}^{\infty} b_k \left[\sum_{j=1}^{\infty} a_j(t-t_o)^j\right]^k . \tag{4.3}$$

Letting

$$\sum_{j=k}^{\infty} c_{jk}(t-t_o)^j = \left[\sum_{j=1}^{\infty} a_j(t-t_o)^j\right]^k \quad \text{for } k\geq 1 \tag{4.4}$$

and interchanging the order of summation in (4.3) leads to

$$t-t_o=\sum_{j=1}^{\infty}(t-t_c)^j \sum_{k=1}^{j} c_{jk}b_k. \tag{4.5}$$

Equating powers of $(t-t_o)$ in (4.5) yields

$$\begin{aligned} b_1&=1/c_{11} \\ b_j&=\Big(\sum_{k=1}^{j-1} c_{jk}b_k\Big)/c_{jj} \end{aligned} \qquad \text{for } j\geq 2 \ . \tag{4.6}$$

Rewriting (4.4) in terms of previously computed coefficients, we find

$$\begin{aligned} \sum_{j=k}^{\infty} c_{jk}(t-t_o)^j &= \sum_{j=k-1}^{\infty} c_{j,k-1}(t-t_o)^j \sum_{j=1}^{\infty} a_j(t-t_o)^j \\ &=\sum_{r=1}^{\infty}\sum_{s=k-1}^{\infty} c_{s,k-1}a_r(t-t_o)^{r+s} \end{aligned} \tag{4.7}$$

$$\text{for } k\geq 2 \ .$$

Substituting $j=r+s$ and interchanging the order of summation yields

$$\sum_{j=k}^{\infty} c_{jk}(t-t_o)^j=\sum_{j=k}^{\infty}(t-t_o)^j \sum_{r=1}^{j-k+1} c_{j-r,k-1}a_r \tag{4.8}$$

$$\text{for } k\geq 2 \ .$$

Finally, equating powers of $(t-t_o)$ yields

$$c_{jk}=\sum_{r=1}^{j-k+1} c_{j-r,k-1}\, a_r \quad \text{for } k\geq 2 \tag{4.9}$$

$$j\geq k$$

Also, note that

$$c_{j1}=a_j \qquad \text{for } j\geq 1 \ . \tag{4.10}$$

The following summarizes the algorithm to find t' such that $f(t')=0$ when the a_j are known, t_o is given sufficiently close to t', and $f_o=f(t_o)$.

$$\begin{aligned}
&1)\quad c_{11}=a_1 \ , \quad b_1=1/c_{11} \\
&2)\quad c_{j1}=a_{j-k+1} \\
&3)\quad c_{jk}=\sum_{r=1} c_{j-r,k-1}\, a_r \quad \text{for } 2\leq k\leq j \\
&4)\quad b_j=\sum_{k=1}^{j-1} c_{jk}b_k/c_{jj}
\end{aligned} \tag{4.11}$$

Repeat 2) thru 4) for j=2,3...

$$5)\quad t'=t_o+\sum_{k=1}^{\infty} b_k(-f_o)^k$$

To illustrate the application of this method, the differential equation for the ninth degree Legendre polynomial was integrated and the zeros of the function computed by series inversion. The results were accurate to the requested precision.

For the computation of the partial derivative of a function of several variables $f(y_1, y_2,\ldots,y_n)$ with respect to y_i, the Taylor series coefficients for the differential system

$$\frac{dy_j}{dt} = \delta_{ij} \qquad j=1,\ldots,n \tag{4.12}$$

are evaluated along with the coefficients for the function f(t). The derivative of f with respect to t may be written as

$$\frac{df}{dt} = \sum_{s=1}^{n} \frac{\partial f}{\partial y_s} \frac{dy_s}{dt} \quad . \tag{4.13}$$

Substituting (4.12) into (4.13), it is clear that the desired partial derivative is the first Taylor coefficient of f. This procedure may be applied to any number of functions and was used to evaluate the Jacobian of the system given in Problem 3. The results of this computation were as accurate as the input data.

V. PL/I Implementation

The program to generate a Fortran subroutine which evaluates recursive Taylor series coefficients for a system of differential equations has been written in PL/I. The PL/I language was chosen, instead of a string processing language like SNOBOL, because PL/I contains an adequate set of string manipulating functions and because of the similarity between PL/I and Fortran statements. Since the PL/I statements are Fortran-like, changes may be incorporated into the processing program to suit individual needs, with greater facility than might otherwise be the case.

In the implementation of [3], the defining system may contain derivatives of arbitrary order and the differentiation operator may appear on the right hand side of the equations. Without a serious loss in generality, the current implementation is restricted to systems of first order differential equations and the differentiation operator may not appear on the right hand side of the equations.

The program reads in the defining system of equations from the PL/I SYSIN data set, and the equations are checked for balance with respect to parentheses, but no determination is yet made as to whether they represent valid expressions. The program then attempts to generate the Fortran subroutine to evaluate the Taylor series coefficients.

It is not a difficult task to add information about the interval of integration, the accuracy required per integration step, etc. to the input definition of the system of equations. This information may then be edited into appropriate driver routines. The next job steps are compilation and execution of the Fortran program. However, these additions are dependent upon the computer installation and upon individual requirements. The PL/I program is written and annotated so that modifications, similar to the ones just mentioned, are fairly straightforward.

The first input card to the PL/I program is a control card containing the words DIFFERENTIAL EQUATIONS, which may appear anywhere in columns 1 to 72. Sequence numbers are permitted in columns 73-80. The word EQUATIONS may be optionally followed by the letters SP or DP, which is a request to generate a single or double precision routine. The default value is single precision. The nomenclature for the i^{th} equation in the differential system is Y(1,I)=f(T,Y), where the first subscript of Y denotes differentiation with respect to the independent variable T. f(T,Y) represents a valid Fortran expression. If it is more convenient to specify the system with a different independent and dependent variable, say R and V, then it is necessary to include (R,V) on the first control card. The differential equations are specified next with a free form

format in columns 1-72. Each differential equation is ended with a semi-colon, except for the last equation, which is terminated with a colon. Any equations which are useful in defining the differential system may be included and are order independent. The next card following the differential equations is a control card containing the words INITIAL VALUES. The initial values are then specified in the same manner as the differential equations.

To illustrate a sample input to the processing program, consider Problem 3 above written as four first order equations. The input data required to specify the construction of a double precision routine may have the following form

```
DIFFERENTIAL EQUATIONS DP(T,U)
U(1,1) = U(3); U(1,2) = U(4);
U(1,3) = U(1) + 2.D0*U(4) - AP*(U(1) + G) - A*(U(1) - GP);
U(1,4) = U(2) - 2.D0*U(3) - U(2)*(AP + A);
G = 1.D0/82.45D0; GP = 1.D0 - G;
A = G/DSQRT((U(1)- GP)**2 + U(2)**2)**3;
AP = GP/DSQRT((U(1)+ G)**2 + U(2)**2)**3:
INITIAL VALUES
U(1) = 1.2D0; U(2) = 0.D0;U(3) = 0.D0;U(4) = -1.04935750983:
```

The generated Fortran routine will have the structure

```
SUBROUTINE COEFF (U, ITSMAX)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION  U(ITSMAX, 1)
Fortran statements necessary to compute the
1 to ITSMAX Taylor coefficients for the equiva-
lent canonical system given U(I), I=1,4.
RETURN
```

```
      ENTRY INITAL (T,U,ITSMAX)
C
C     Initialization of all Taylor coefficients
C     to zero followed by the assignment of the
C     initial values specified as input data.
C
      RETURN
      END
```

With the above routine and the two Fortran routines listed in Appendix B, edited in the appropriately indicated places, a complete Fortran program for the numerical integration of the sample problem may be developed.

Standard output from the processing program contains listings of the defining differential equations and the generated Fortran routine. Since it is necessary to have some measure of the computer time required per pass through the COEFF routine in order to properly assess the effectiveness of the Taylor series method compared to other popular methods for solving differential equations, an operations count in terms of additions and multiplications is also printed.

The process by which the Fortran routine is generated is very similar to the way a compiler generates assembler language routines. For a complete description of the algorithm see [3]. The differential system is reduced to canonical form, which is the representation of the system in terms of the elementary operations of + - * /. The

decomposition is accomplished by the method of bounded context translation [14]. The next step consists of an elimination of redundant operations from the canonical system. After the system has been optimized, a tree search is performed to determine the computational order. For some equations, it may be desirable to examine a number of the intermediate quantities in this process. Coding DEBUG in the PARM field of the processing programs EXEC statement will produce this listing. For the Riccati equation, $y' = y^2 + 3t^2$, the DEBUG listing has the form given in Appendix D.

RMAT is the procedure which performs the decomposition of the differential system. LEVEL denotes the current level of recursive calls to the procedure. The integer K denotes the element in the equation being scanned. TYPE is an integer representing the $K^{\underline{th}}$ element. Table V-A is a listing of the correspondence between the integers and the elements. The E in (C,O,V|E) denotes the print mode that lists the input equation to the procedure. The equation is enclosed by the delimiters #$. C,O,V represents the print mode that lists the $K^{\underline{th}}$ element which is either a constant, operator, or variable. If the $K^{\underline{th}}$ element is a constant, it is replaced by #ℓ. The integer ℓ designates the position of this constant in a tabulation of all constants that appear in the differential system. The constant table is

TABLE V-A

TYPE	ELEMENT
-1	constant
0	variable
1	+
2	-
3	*
4	/
5	=
6	(
7	)
8	# (left delimiter)
9	$ (right delimiter)
10	% (function specification)
11	**

listed after the optimized canonical form.

The heading on the right indicates entries into the recurrence matrix, where the canonical system is eventually stored. R represents the row of the matrix, OP the operation, and A(1), A(2) the two possible arguments. A(3) is the name associated with this operation. If there is no external name associated with this operation, the name is generally represented as ?r, where r indicates the row in the matrix storing the result of the operation. The symbol $ in the recurrence matrix is used for the composite operation = . The first differential equation processed is the one for the independent variable, which makes the system autonomous. After the last equation has been processed, the complete recurrence matrix is listed both before and after it has been optimized. As mentioned earlier, the constant table is listed at this point.

The next step involves searching the recurrence matrix to initialize the matrix D described in [3]. The D matrix is used to determine the computational order of evaluation of the coefficients. The dimension of the matrix is the number of rows in the recurrence matrix. Briefly, starting with the result of the operation for a given row in the recurrence matrix, the arguments of the operation are traced backwards thru the recurrence matrix to ascertain

TABLE V-B

%	Function
1	exp
2	$\log_{10}$
3	$\log_e$
4	sin
5	cos
6	tan
7	sinh
8	cosh
9	tanh
10	sqrt

their dependence upon other operations. The DMAT ENTRY statement lists the row currently being initialized, and finally the entire D matrix is listed. To aid in an interpretation of the recurrence matrix, a table is constructed showing the correspondence between this matrix and the set of dependent variables in the canonical system. An integer pair, ij, in this table, indicates that the result of the operation in the $i^{\underline{th}}$ row of the recurrence matrix is the $j^{\underline{th}}$ dependent variable.

In the reduction to canonical form, special functions which appear may cause their defining differential equations to be appended to the differential system. In this implementation, the special functions are left in the reduced system and the corresponding coefficients for these functions are hard coded in the program generating routine. The symbol %j, where j represents an integer constant, is used to represent functions in the recurrence matrix. Table V-B shows the correspondence between the integers j and the functions they represent.

This completes the description of the intermediate quantities required in the Fortran COEFF routine construction. The listings should be useful in debugging any malfunctioning of the processing program for a given differential system.

REFERENCES

1. Collatz, L. The Numerical Treatment of Differential Equations, Springer-Verlag, Berlin, 1960.

2. Moore, R.E. Interval Analysis, Prentice-Hall, Englewood Cliffs, N.J., 1966, pp. 107-118.

3. Barton, D., Willers, I.M., and Zahar, R.V.M. "The Automatic Solution of Systems of Ordinary Differential Equations by the Method of Taylor Series", The Computer Journal, V. 14, 1971, pp. 243-248.

4. System/360 Scientific Subroutine Package Programmers Manual, IBM, subroutine RKGS.

5. Bulirsch, R., and Stoer, J. "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods", Numerische Mathematik, V. 8, 1966, pp. 1-13.

6. Barton, D., Willers, I.M., and Zahar, R.V.M. "Taylor Series Methods for Ordinary Differential Equations - An Evaluation", Proc. Math. Software Symposium, Purdue Univ., 1970, pp. 369-390.

7. Eiserike, H., and Silver, A. "A Study of Nordsieck-Type Predictor-Corrector Methods", NASA GSFC X-641-70-199, Revised 1971.

8. Henrici, P. Elements of Numerical Analysis, Wiley, New York, 1964, pp. 51-52.

9. Gear, C.W. "The Automatic Integration of Ordinary Differential Equations", Comm. ACM, V. 14, 1971, pp. 176-190.

10. Estes, R.H., and Lancaster, E.R. "Optimized Computation with Recursive Power Series Integration", NASA GSFC X-643-68-80, 1968.

11. Abramowitz, M. "Coulomb Wave Functions" in Handbook of Mathematical Functions, National Bureau of Standards, 1965, pp. 537-554.

12. Liniger, W., and Odeh, F. "A-Stable, Accurate Averaging of Multistep Methods for Stiff Differential Equations", IBM J. Res. Develop. V. 16, 1972, pp. 335-348.

13. Knopp, K. Infinite Sequences and Series, Dover, New York, 1956, pp. 119-124.

14. Graham, R. "Bounded Context Translation", AFIPS-SJCC, V. 25, 1964, pp. 17-29.

APPENDIX A

```
TAYLOR: PROC(PARM) OPTIONS(MAIN);                                        00000100
/*********************************************************************** 00000200
 * DRIVER PROCEDURE USED TO CONSTRUCT A FORTRAN SUBROUTINE WHICH      * 00000300
 * EVALUATES THE RECURSIVE TAYLOR COEFFICIENTS DERIVED FROM A         * 00000400
 * SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS. FOR A DESCRIPTION OF    * 00000500
 * THE ALGORITHM SEE 'THE AUTOMATIC SOLUTION OF SYSTEMS OF            * 00000600
 * ORDINARY DIFFERENTIAL EQUATIONS BY THE TAYLOR SERIES METHOD',      * 00000700
 * BY D. BARTON ET AL. COMPUTER JOURNAL V 14 (1971) PP. 243-248       * 00000800
 ***********************************************************************/00000900
     DCL                                                                 00001000
      BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                       00001100
      BREAKF ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR),                 00001200
      CODE ENTRY(CHAR(*) VAR,BIT(1),BIT(1)),                             00001300
      COUNT ENTRY(CHAR(*) VAR,CHAR(*) VAR) RETURNS(BIN FIXED),           00001400
      SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);       00001500
     DCL                                                                 00001600
      (CS,WS) CHAR(400) VAR EXT,DVRBL CHAR(4) VAR EXT,                   00001700
      IVRBL CHAR(4) VAR EXT,R( 500,4) CHAR(15) VAR,                      00001800
      RMAX BIN FIXED INIT(0),D(*,*) BIN FIXED CTL,KFMAX EXT INIT(0),     00001900
      ERROR BIN FIXED,CC BIN FIXED INIT(1),IED EXT,                      00002000
      DEBUG BIT(1) EXT,NEQ EXT INIT(0),KO BIT(1),NSGMA EXT INIT(0),      00002100
      FL FILE OUTPUT,SN CHAR(4) VAR,CB CHAR(15) VAR;                     00002200
     DCL                                                                 00002300
      (NMUL,NADD,NMTS,NMTL,NATS,NATL) INIT(0) EXT,PARM CHAR(100) VAR,    00002400
      LBLA(3) LABEL INIT(LW,LW,LD),LBLB(3) LABEL INIT(LX,LD,LD),         00002500
      CST(100) CHAR(25) VAR EXT,IC EXT INIT(1),NEQTNS EXT;               00002600
/*                                                                      */00002700
     CALL STINT;                                                         00002800
     IF INDEX(PARM,'DEBUG')^=0 THEN DEBUG='1'B;                          00002900
/* READ IN THE SYSTEM OF EQUATIONS */                                   00003000
     CST(1)='0.5';                                                       00003100
     CALL INPUT(ERROR,CC);                                               00003200
     IF IED=0 THEN CST(1)='0.5'; ELSE CST(1)='0.5D0';                    00003300
     GO TO LBLA(ERROR);                                                  00003400
LW:  NEQTNS=COUNT(CS,'Y(1,')+1;                                          00003500
/* INSERT DIFFERENTIAL EQUATION FOR THE INDEPENDENT VARIABLE TO MAKE    00003600
   THE SYSTEM AUTONOMOUS */                                             00003700
     CS='Y(1,'||BFDC(NEQTNS)||')=1.0'||';'||CS||'#';                     00003800
/* READ IN THE INITIAL VALUES */                                        00003900
     CC=2;                                                               00004000
     CALL INPUT(ERROR,CC);                                               00004100
     CS=CS||DVRBL||'('||BFDC(NEQTNS)||')='||IVRBL||';';                  00004200
     GO TO LBLB(ERROR);                                                  00004300
LX:  IPASS=1;                                                            00004400
/* FACTOR EACH DIFFERENTIAL EQUATION INTO ELEMENTARY OPERATIONS */      00004500
LY:  DO WHILE(SUBSTR(CS,1,1)^='#');                                      00004600
        NEQ=NEQ+1;                                                       00004700
        CALL BREAKF(CS,';',WS);                                          00004800
        CALL RMAT('#'||WS||'S',R,RMAX,KO);                               00004900
```

```
         IF NEQ=1 THEN R(RMAX,4)=IVRBL;                                  00005000
      END;                                                               00005100
      IF ¬DEBUG THEN GO TO LZ;                                           00005200
      PUT EDIT('RECURRENCE MATRIX') (SKIP(1),X(60),A);                   00005300
      DO I=1 TO RMAX;                                                    00005400
         PUT EDIT(I,R(I,1),R(I,2),R(I,3),R(I,4)) (SKIP,X(60),            00005500
          F(2),X(1),A(2),X(5),3 A(15));                                  00005600
      END;                                                               00005700
/* ELIMINATE REDUNDANT OPERATIONS FROM THE RECURRENCE MATRIX */           00005800
LZ:   CALL OPTMZE(R,RMAX);                                               00005900
      IF ¬DEBUG THEN GO TO LB;                                           00006000
      PUT EDIT('OPTIMIZED RECURRENCE MATRIX') (SKIP(1),X(60),A);         00006100
LA:   DO I=1 TO RMAX;                                                    00006200
         PUT EDIT(I,R(I,1),R(I,2),R(I,3),R(I,4)) (SKIP,X(60),F(2),       00006300
          X(1),A(2),X(5),3 A(15));                                       00006400
      END;                                                               00006500
LB:   IF ¬DEBUG THEN GO TO LBC;                                          00006600
      PUT EDIT('CONSTANT TABLE') (SKIP(2),A);                            00006700
      DO I=1 TO IC;                                                      00006800
         PUT EDIT('#',I,'=',CST(I)) (COLUMN(MOD(I-1,4)*29+1),            00006900
          A,F(2),A,A);                                                   00007000
      END;                                                               00007100
LBC:  ALLOCATE D(RMAX,RMAX);                                             00007200
/* GENERATE THE MATRIX D WHICH IS USED TO DETERMINE THE ORDER IN WHICH   00007300
   THE TAYLOR COEFFICIENTS ARE COMPUTED */                               00007400
      CALL DMAT(R,RMAX,D);                                               00007500
      IF ¬DEBUG THEN GO TO LC;                                           00007600
      PUT EDIT('D MATRIX') (SKIP(2),A);                                  00007700
      PUT EDIT(((I,',',J,D(I,J) DO J=1 TO RMAX) DO I=1 TO RMAX))         00007800
       (SKIP,8 (F(2),A,F(2),F(4),X(5)));                                 00007900
/* GENERATE THE FORTRAN ROUTINE TO COMPUTE THE TAYLOR COEFFICIENTS */     00008000
      CALL CODE(R,RMAX,D,KO);                                            00008100
      IF ¬KO THEN GO TO LD;                                              00008200
LC:   FREE D;                                                            00008300
      IPASS=IPASS+1;                                                     00008400
      IF IPASS>2 THEN GO TO LD;                                          00008500
      CALL CODE('C','0'B,'1'B);                                          00008600
      CALL CODE('      ENTRY INITAL('||IVRBL||','||DVRBL||',ITSMAX)',    00008700
       '0'B,'1'B);                                                       00008800
      CALL CODE('      DO 2001 ITS=1,ITSMAX','0'B,'1'B);                 00008900
      CALL CODE('      DO 2001 IXV=1,'||BFDC(RMAX),'0'B,'1'B);           00009000
      CALL CODE('2001  Y(ITS,IXV=0.0','0'B,'1'B);                        00009100
      DO WHILE(CS¬='');                                                  00009200
         CALL BREAKF(CS,':',WS);                                         00009300
         CALL SPAN(WS,'(',')',SN);                                       00009400
         CALL BREAKF(WS,')',CB);                                         00009500
         CALL CODE(DVRBL||'(1,'||SN||')'||WS,'1'B,'0'B);                 00009600
      END;                                                               00009700
      CALL CODE(DVRBL||'(2,'||BFDC(NEQTNS)||')=1.0','0'B,'0'B);          00009800
```

```
      CALL CODE('       RETURN',''0'B,''1'B);                                  00009900
      CALL CODE('       END',''0'B,''1'B);                                     00010000
      PUT EDIT('OPERATION COUNT (OC) FOR ONE PASS THRU THE COEFF '            00010100
       ||'ROUTINE') (SKIP(3),A);                                              00010200
      PUT EDIT('ITSMAX - THE NUMBER OF COEFFICIENTS COMPUTED')                00010300
       (SKIP(2),A);                                                           00010400
      PUT EDIT('AS      - AN ADDITION OR SUBTRACTION') (SKIP,A);              00010500
      PUT EDIT('MD      - A MULTIPLICATION OR DIVISION') (SKIP,A);            00010600
      WS='OC = ('||BFDC(NADD)||' + ('||BFDC(NATL)||' + '||BFDC(NATS)||        00010700
       '*ITSMAX)*ITSMAX/2)*AS + ('||BFDC(NMUL)||' + ('||BFDC(NMTL)||          00010800
       ' + '||BFDC(NMTS)||'*ITSMAX)*ITSMAX/2*MD';                             00010900
      PUT EDIT(WS) (SKIP(2),A);                                               00011000
LD: END TAYLOR;                                                               00011100
```

```
* PROCESS;
 TAYLOR: PROC(PARM) OPTIONS(MAIN);                                        00000100
 /*************************************************************** 00000200
  * DRIVER PROCEDURE USED TO CONSTRUCT A FORTRAN SUBROUTINE WHICH     * 00000300
  * EVALUATES THE RECURSIVE TAYLOR COEFFICIENTS DERIVED FROM A        * 00000400
  * SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS. FOR A DESCRIPTION OF   * 00000500
  * THE ALGORITHM SEE 'THE AUTOMATIC SOLUTION OF SYSTEMS OF           * 00000600
  * ORDINARY DIFFERENTIAL EQUATIONS BY THE TAYLOR SERIES METHOD',     * 00000700
  * BY D. BARTON ET AL. COMPUTER JOURNAL V 14 (1971) PP. 243-248      * 00000800
  ***************************************************************/00000900
      DCL                                                               00001000
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                     00001100
       BREAKF ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR),               00001200
       CODE ENTRY(CHAR(*) VAR,BIT(1),BIT(1)),                           00001300
       COUNT ENTRY(CHAR(*) VAR,CHAR(*) VAR) RETURNS(BIN FIXED),         00001400
       SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);     00001500
      DCL                                                               00001600
       (CS,WS) CHAR(400) VAR EXT,DVRBL CHAR(4) VAR EXT,                 00001700
       IVRBL CHAR(4) VAR EXT,R( 500,4) CHAR(15) VAR,                    00001800
       RMAX BIN FIXED INIT(0),D(*,*) BIN FIXED CTL,KFMAX EXT INIT(0),   00001900
       ERROR BIN FIXED,CC BIN FIXED INIT(1),IED EXT,                    00002000
       DEBUG BIT(1) EXT,NEQ EXT INIT(0),KO BIT(1),NSGMA EXT INIT(0),    00002100
       FL FILE OUTPUT,SN CHAR(4) VAR,CB CHAR(15) VAR;                   00002200
      DCL                                                               00002300
       (NMUL,NADD,NMTS,NMTL,NATS,NATL) INIT(0) EXT,PARM CHAR(100) VAR,  00002400
       LBLA(3) LABEL INIT(LW,LW,LD),LBLB(3) LABEL INIT(LX,LD,LD),       00002500
       CST(100) CHAR(25) VAR EXT,IC EXT INIT(1),NEQTNS EXT;             00002600
 /*                                                                     */00002700
      CALL STINT;                                                       00002800
      IF INDEX(PARM,'DEBUG')¬=0 THEN DEBUG='1'B;                        00002900
 /* READ IN THE SYSTEM OF EQUATIONS */                                  00003000
      CST(1)='0.5';                                                     00003100
      CALL INPUT(ERROR,CC);                                             00003200
      IF IED=0 THEN CST(1)='0.5'; ELSE CST(1)='0.500';                  00003300
      GO TO LBLA(ERROR);                                                00003400
 LW:  NEQTNS=COUNT(CS,'Y(1,')+1;                                        00003500
 /* INSERT DIFFERENTIAL EQUATION FOR THE INDEPENDENT VARIABLE TO MAKE   00003600
    THE SYSTEM AUTONOMOUS */                                            00003700
      CS='Y(1,'||BFDC(NEQTNS)||')=1.0'||';'||CS||'#';                   00003800
 /* READ IN THE INITIAL VALUES */                                       00003900
      CC=2;                                                             00004000
      CALL INPUT(ERROR,CC);                                             00004100
      CS=CS||DVRBL||'('||BFDC(NEQTNS)||')='||IVRBL||';';                00004200
      GO TO LBLB(ERROR);                                                00004300
 LX:  IPASS=1;                                                          00004400
 /* FACTOR EACH DIFFERENTIAL EQUATION INTO ELEMENTARY OPERATIONS */     00004500
 LY:  DO WHILE(SUBSTR(CS,1,1)¬='#');                                    00004600
         NEQ=NEQ+1;                                                     00004700
         CALL BREAKF(CS,':',WS);                                        00004800
```

```
         CALL RMAT('#'||WS||'$',R,RMAX,KO);                                00004900
         IF NEQ=1 THEN R(RMAX,4)=IVRBL;                                    00005000
      END;                                                                 00005100
      IF ¬DEBUG THEN GO TO LZ;                                             00005200
      PUT EDIT('RECURRENCE MATRIX') (SKIP(1),X(60),A);                     00005300
      DO I=1 TO RMAX;                                                      00005400
         PUT EDIT(I,R(I,1),R(I,2),R(I,3),R(I,4)) (SKIP,X(60),              00005500
          F(2),X(1),A(2),X(5),3 A(15));                                    00005600
      END;                                                                 00005700
/* ELIMINATE REDUNDANT OPERATIONS FROM THE RECURRENCE MATRIX */            00005800
LZ:   CALL OPTMZE(R,RMAX);                                                 00005900
      IF ¬DEBUG THEN GO TO LA;                                             00006000
      PUT EDIT('OPTIMIZED RECURRENCE MATRIX') (SKIP(1),X(60),A);           00006100
LA:   DO I=1 TO RMAX;                                                      00006200
         PUT EDIT(I,R(I,1),R(I,2),R(I,3),R(I,4)) (SKIP,X(60),F(2),         00006300
          X(1),A(2),X(5),3 A(15));                                         00006400
      END;                                                                 00006500
LB:   IF ¬DEBUG THEN GO TO LBC;                                            00006600
      PUT EDIT('CONSTANT TABLE') (SKIP(2),A);                              00006700
      DO I=1 TO IC;                                                        00006800
         PUT EDIT('#',I,'=',CST(I)) (COLUMN(MOD(I-1,4)*29+1),              00006900
          A,F(2),A,A);                                                     00007000
      END;                                                                 00007100
LBC:  ALLOCATE D(RMAX,RMAX);                                               00007200
/* GENERATE THE MATRIX D WHICH IS USED TO DETERMINE THE ORDER IN WHICH    00007300
   THE TAYLOR COEFFICIENTS ARE COMPUTED */                                 00007400
      CALL DMAT(R,RMAX,D);                                                 00007500
      IF ¬DEBUG THEN GO TO LC;                                             00007600
      PUT EDIT('D MATRIX') (SKIP(2),A);                                    00007700
      PUT EDIT(((I,',',J,D(I,J) DO J=1 TO RMAX) DO I=1 TO RMAX))           00007800
       (SKIP,8 (F(2),A,F(2),F(4),X(5)));                                   00007900
/* GENERATE THE FORTRAN ROUTINE TO COMPUTE THE TAYLOR COEFFICIENTS */      00008000
      CALL CCGE(R,RMAX,D,KO);                                              00008100
      IF ¬KO THEN GO TO LD;                                                00008200
LC:   FREE D;                                                              00008300
      IPASS=IPASS+1;                                                       00008400
      IF IPASS>2 THEN GO TO LD;                                            00008500
      CALL CODE('C','0'B,'1'B);                                            00008600
      CALL CODE('      ENTRY INITAL('||IVRBL||','||DVRBL||',ITSMAX)',      00008700
       '0'B,'1'B);                                                         00008800
      CALL CODE('      DO 2001 ITS=1,ITSMAX','0'B,'1'B);                   00008900
      CALL CODE('      DO 2001 IXV=1,'||BFDC(RMAX),'0'B,'1'B);             00009000
      CALL CODE('2001  Y(ITS,IXV)=0.0','0'B,'1'B);                         00009100
      DO WHILE(CS¬='');                                                    00009200
         CALL BREAKF(CS,':',WS);                                           00009300
         CALL SPAN(WS,'(',')',SN);                                         00009400
         CALL BREAKF(WS,')',CB);                                           00009500
         CALL CODE(DVRBL||'(I,'||SN||')'||WS,'1'B,'0'B);                   00009600
      END;                                                                 00009700
```

```
      CALL CODE(DVRBL||'(2,'||BFDC(NFCTNS)||')=1.0','0'B,'0'B);          00009800
      CALL CODE('       RETURN','0'B,'1'B);                              00009900
      CALL CODE('       END','0'B,'1'B);                                 00010000
      PUT EDIT('OPERATION COUNT (OC) FOR ONE PASS THRU THE COEFF '       00010100
       ||'ROUTINE') (SKIP(3),A);                                         00010200
      PUT EDIT('ITSMAX - THE NUMBER OF COEFFICIENTS COMPUTED')           00010300
       (SKIP(2),A);                                                      00010400
      PUT EDIT('AS      - AN ADDITION OR SUBTRACTION') (SKIP,A);         00010500
      PUT EDIT('MD      - A MULTIPLICATION OR DIVISION') (SKIP,A);       00010600
      WS='OC = ('||BFDC(NADD)||' + ('||BFDC(NATL)||' + '||BFDC(NATS)||   00010700
       '*ITSMAX)*ITSMAX/2)*AS + ('||BFDC(NMUL)||' + ('||BFDC(NMTL)||     00010800
       ' + '||BFDC(NMTS)||'*ITSMAX)*ITSMAX/2*MD';                        00010900
      PUT EDIT(WS) (SKIP(2),A);                                          00011000
LD: END TAYLOR;                                                          00011100
```

```
* PROCESS;                                                                    00000100
 CODE: PROC(STRING,SUM,CMMNT) RECURSIVE;                                     00000200
 /* PROCEDURE TRANSFORMS THE INPUT 'STRING' INTO FORTRAN CARD IMAGES */      00000300
       DCL                                                                   00000400
        EXTRACT ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR),                  00000500
        LIBF ENTRY(CHAR(*) VAR,BIN FIXED),                                   00000600
        SIGMA ENTRY(CHAR(*) VAR);                                            00000700
       DCL ST CHAR(400) VAR,(SUM,CMMNT) BIT(1),STRING CHAR(*) VAR,           00000800
        SN CHAR(7) VAR,FL FILE OUTPUT EXT,IED EXT;                           00000900
       DCL NSGMA STATIC BIN FIXED EXT,SYSA BIT(1) EXT,IC INIT(1),            00001000
         SQN BIN FIXED STATIC INIT(10000), EIS(3) CHAR(8) VAR EXT;           00001100
 /*                                                                          */ 00001200
       IF ¬SYSA|LENGTH(STRING)<11 THEN GO TO LA;                             00001300
       SN=SUBSTR(STRING,7,5);                                                00001400
       IF SN='SUBRO' | SN='BLOCK' THEN PUT PAGE;                             00001500
 LA:   IF ¬SUM THEN GO TO LC;                                                00001600
 LB:   CALL EXTRACT(STRING,EIS(3),ST);                                       00001700
       IF ST='' THEN GO TO LC;                                               00001800
       CALL SIGMA(ST);                                                       00001900
       GO TO LB;                                                             00002000
 LC:   IF ¬CMMNT THEN GO TO LD;                                              00002100
       SQN=SQN+100;                                                          00002200
       PUT FILE(FL) EDIT(STRING,'000'||SQN)(SKIP,A,COLUMN(73),A,F(5));       00002300
       IF SYSA THEN PUT EDIT(STRING,'000'||SQN)(SKIP,A,COLUMN(73),A,F(5));   00002400
       RETURN;                                                               00002500
 LD:   SN='      ';                                                          00002600
       IF ¬CMMNT THEN CALL LIBF(STRING,IED);                                 00002700
       DO I=1 TO 6 WHILE(VERIFY(SUBSTR(STRING,I,1),'0123456789 ')=0);END;    00002800
       IF I¬=1 THEN SN=SUBSTR(SUBSTR(STRING,1,I-1) || SN,1,6);               00002900
       STRING=SUBSTR(STRING,I);                                              00003000
       LS=LENGTH(STRING);                                                    00003100
       DO I=1 TO LS BY 65;                                                   00003200
          ST=SN || SUBSTR(STRING,IC,MIN(65,LS-IC+1));                        00003300
          SQN=SQN+100;                                                       00003400
          PUT FILE(FL) EDIT (ST,'000'||SQN) (SKIP,A,COLUMN(73),A,F(5));      00003500
          IF SYSA THEN PUT EDIT(ST,'000'||SQN) (SKIP,A,COLUMN(73),A,F(5));   00003600
          IC=IC+65;                                                          00003700
          IF I=1 THEN SN='      .  ';                                        00003800
       END;                                                                  00003900
 END CODE;                                                                   00004000
```

```
* PROCESS;                                                                    00000100
 CODE: PROC(R,RMAX,D,KO);                                                    00000200
 /* PROCEDURE GENERATES THE FORTRAN TAYLOR COEFFICIENT ROUTINE */            00000300
      DCL                                                                    00000400
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                          00000500
       DFTC ENTRY(BIN FLOAT(53),BIN FIXED)  RETURNS(CHAR(50) VAR),           00000600
       BREAKF ENTRY(CHAR(*) VAR,CHAR(1),CHAR(*) VAR),                        00000700
       CODE ENTRY(CHAR(*) VAR,BIT(1),BIT(1)),                                00000800
       COL4 ENTRY RETURNS(BIN FIXED),                                        00000900
       FUDGE ENTRY RETURNS(BIT(1)),                                          00001000
       OP ENTRY(CHAR(1)) RETURNS(BIN FIXED),                                 00001100
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1)),            00001200
       SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);          00001300
      DCL                                                                    00001400
       R(*,*) CHAR(*) VAR,D(*,*) BIN FIXED,DR(*) BIT(1) CTL,KO BIT(1),       00001500
       CTBL(*) CHAR(15) VAR CTL,DVRBL CHAR(4) VAR EXT,(DMAX,RMAX)            00001600
       BIN FIXED,(M(*),C(*)) BIN FIXED CTL,FTBL(100,2) BIN FIXED EXT,        00001700
       L3L(11) LABEL INIT(PMS,PMS,MPY,DVD,EOL,ERR,                           00001800
       ERR,ERR,INT,FA,EXP),CST(100) CHAR(25) VAR EXT,CO CHAR(1),IED EXT,     00001900
       L(4) LABEL,OC(10) CHAR(1) EXT,WS CHAR( 400) VAR EXT,NEQTNS EXT;       00002000
      DCL                                                                    00002100
       LEPMAX(3) INIT(3,4,2),LEP INIT(0),CH CHAR(1),                         00002200
       IVFN CHAR(10) VAR,IVRBL CHAR(4) VAR EXT,                              00002300
       DEBUG BIT(1) EXT,DT(*,*) BIN FIXED CTL,                               00002400
       C48 CHAR(1) EXT,(NMUL,NADD,NMTS,NMTL,NATS,NATL) EXT,                  00002500
       IQ BIT(1) INIT('1'B),ACS CHAR(50) VAR,EPLG CHAR(200) VAR;             00002600
      DCL                                                                    00002700
       (ARG(2) CHAR(25) VAR,(CA3,ZERO) BIT(1),(LHS,LHSARG) CHAR(15) VAR,     00002800
       (LP(2),RP(2),UO(2)) CHAR(1) VAR,KCMAX,KFMAX,TSS CHAR(5) VAR           00002900
       ) EXT;                                                                00003000
 /*                                                                          */00003100
      ZERO='1'B;                                                             00003200
      EPLG=                                                                  00003300
       'SUBROUTINE COEFF('||DVRBL||',ITSMAX);IMPLICIT REAL*'||C48||          00003400
       '(A-H,O-Z);'||'DIMENSION '||DVRBL||'(ITSMAX,1);1000 DO 2000 '||       00003500
       'ITS=2,ITSMAX;ITSM1=ITS-1;ITSP1=ITS+1;FITSM1=FLOAT(ITSM1);'||         00003600
       '2000CONTINUE;RETURN;';                                               00003700
      IVFN=DVRBL||'('||BFDC(NEQTNS)||')';                                    00003800
      DMAX=DIM(D,1);                                                         00003900
      ALLOCATE DR(DMAX),CTBL(RMAX),M(DMAX),C(DMAX),DT(RMAX,RMAX);            00004000
      DR='0'B;  DT=D;                                                        00004100
 /* CORRESPONDENCE TABLE BETWEEN R(I,4) & Y(J) */                           00004200
      KC=NEQTNS-1;                                                           00004300
      DO I=1 TO RMAX;                                                        00004400
         IF SUBSTR(R(I,4),1,MIN(LENGTH(R(I,4)),LENGTH(DVRBL)))=DVRBL         00004500
          THEN CALL SPAN(R(I,4),'(',')',CTBL(I));                          00004600
          ELSE DO; KC=KC+1; CTBL(I)=BFDC(KC); END;                         00004700
      END;                                                                   00004800
      IF DEBUG                                                               00004900
```

```
     THEN DO; PUT EDIT('CORRESPONDENCE BETWEEN RECURRENCE MATRIX ROWS '00005000
               ||'AND THE Y ARRAY') (SKIP(2),A);                          00005100
              PUT EDIT((I,CTBL(I) DO I=1 TO RMAX)) (SKIP,12 (F(3),X(1)),00005200
               A(3),X(4)));                                               00005300
          END;                                                            00005400
/* EVALUATION OF THE SET M */                                            00005500
     KM=0;                                                               00005600
     DO J=1 TO DMAX;                                                     00005700
        DO I=1 TO DMAX;                                                  00005800
           IF DP(I) THEN GO TO LB;                                       00005900
           IF DT(I,J)>2 THEN GO TO LC;                                   00006000
LB:     END;                                                             00006100
        GO TO LE;                                                        00006200
LC:     KM=KM+1; M(KM)=J;                                                00006300
LE:  END;                                                                00006400
     IF KM=0 THEN GO TO LEA;                                             00006500
     PUT EDIT('** PROLOG IS NOT CURRENTLY IMPLEMENTED **') (SKIP(2),A);00006600
     KO='0'B;                                                            00006700
     RETURN;                                                             00006800
LEA: PUT PAGE EDIT('** LISTING OF THE GENERATED FORTRAN ROUTINE **')     00006900
      (A); PUT SKIP;                                                     00007000
LF:  LEP=LEP+1;                                                          00007100
     DO I=1 TO LEPMAX(LEP);                                              00007200
        CALL BREAKF(EPLG,';',WS);                                        00007300
        CALL CODE(WS,'0'B,'0'B);                                         00007400
     END;                                                                00007500
     IF EPLG¬='' THEN GO TO LG;                                          00007600
     FREE DR,CTBL,M,C,DT;                                                00007700
     RETURN;                                                             00007800
/* EVALUATION OF THE SET C */                                           00007900
LG:  KC=0;                                                               00008000
     IF ZERO THEN TSS='(1,'; ELSE TSS='(ITS,';                           00008100
     DO I=1 TO DMAX;                                                     00008200
        IF DR(I) THEN GO TO LI;                                          00008300
        IF ¬ZERO & IO                                                    00008400
        THEN DO; IF R(I,1)='S' THEN DO; KCMAX=I; GO TO LK; END;          00008500
                 GO TO LI;                                               00008600
             END;                                                        00008700
        DO J=1 TO DMAX;                                                  00008800
           IF DT(I,J)>0 THEN GO TO LI;                                   00008900
        END;                                                             00009000
        KC=KC+1; KCMAX=I; C(KC)=I;                                       00009100
LI:  END;                                                                00009200
     IF IO &¬ZERO THEN DO; IO='0'B; GO TO LG; END;                       00009300
     IF KC>0 THEN GO TO LK;                                              00009400
     PUT EDIT('** EQUATIONS ARE NOT WELL POSED **') (SKIP(2),A);         00009500
     KO='0'B;                                                            00009600
     PUT EDIT('D MATRIX') (SKIP(2),A);                                   00009700
     PUT EDIT(((I,'.',J,D(I,J) DO J=1 TO RMAX) DO I=1 TO RMAX))          00009800
```

```
      (SKIP,8 (F(2),A,F(2),F(4),X(5)));                                    00009900
     IF DEBUG                                                              00010000
     THEN PUT EDIT(((I,',',J,DT(I,J) DO J=1 TO RMAX) DO I=1 TO RMAX))      00010100
          (SKIP,8 (F(2),A,F(2),F(4),X(5)));                                00010200
     RETURN;                                                               00010300
LK:  LHSARG=CTBL(CCL4(R(KCMAX,4),R,RMAX));                                 00010400
     LHS=DVRBL||TSS||LHSARG||'=';                                          00010500
     CA3=(SUBSTR(R(KCMAX,3),1,1)='#');                                     00010600
     KA=0; UO,LP,RP='';                                                    00010700
     IF R(KCMAX,1)='**'                                                    00010800
     THEN DO; NOP=11; CO=' '; END;                                         00010900
     ELSE DO; NOP=OP(R(KCMAX,1)); CO=CC(NOP); END;                         00011000
     DO I=1 TO 2;                                                          00011100
        IF R(KCMAX,1)='%' & I=1 THEN GO TO LL;                             00011200
        IF SUBSTR(R(KCMAX,I+1),1,1)='#'                                    00011300
         THEN DO; KA=KA+I;                                                 00011400
                  IF NOP<3 & ¬ZERO                                         00011500
                  THEN DO; ARG(I)=''; IF NOP=1|I=2 THEN CO=''; END;        00011600
                  ELSE ARG(I)=CST(SUBSTR(R(KCMAX,I+1),2));                 00011700
              END;                                                         00011800
        ELSE DO; IF R(KCMAX,I+1)=IVFN                                      00011900
                  THEN ARG(I)=CTBL(COL4(IVRBL,R,RMAX));                    00012000
                  ELSE ARG(I)=CTBL(CCL4(R(KCMAX,I+1),R,RMAX));             00012100
                  CH=SUBSTR(R(KCMAX,I+1),1,1);                             00012200
                  IF CH='+' | CH='-'                                       00012300
                  THEN DO; UO(I)=CH; LP(I)='('; RP(I)=')'; END;            00012400
              END;                                                         00012500
LL:  END;                                                                  00012600
     IF KA=0 THEN KA=4;                                                    00012700
     IF NOP=9 THEN GO TO INT;                                              00012800
     IF NOP=11 THEN GO TO EXP;                                             00012900
     IF NOP=5                                                              00013000
     THEN DO; IF ¬ZERO THEN GO TO LT; LHS=''; LP(2),RP(2)=''; END;         00013100
     GO TO L(KA);                                                          00013200
L(1):IF NOP=4                                                              00013300
     THEN WS='-SIGMA(IXV=2,ITS:'||DVRBL||'(IXV,'||ARG(2)||')*'||DVRBL||   00013400
          '(ITSP1-IXV,'||LHSARG||')))/'||DVRBL||'(1,'||ARG(2)||')';        00013500
      ELSE WS=ARG(1)||CO||LP(2)||UO(2)||DVRBL||TSS||ARG(2)||')'||RP(2);    00013600
     GO TO L34;                                                            00013700
L(2): WS=UO(1)||DVRBL||TSS||ARG(1)||')'||CO||LP(2)||ARG(2)||RP(2);         00013800
     GO TO L34;                                                            00013900
L(3): IF ¬ZERO THEN GO TO LT;                                              00014000
     WS=LHS||UO(1)||ARG(1)||CO||LP(2)||UO(2)||ARG(2)||RP(2);               00014100
L34: IF NOP<3 THEN NADD=NADD+1;                                            00014200
     IF NOP<5 & NOP>2 THEN NMUL=NMUL+1;                                    00014300
     GO TO LS;                                                             00014400
L(4): GO TO LBL(NOP);                                                      00014500
FN:  IF ¬FUDGE(CTBL,DR,FTBL,R,RMAX,DT) THEN RETURN; ELSE GO TO LU;         00014600
ERR: PUT EDIT('** ILLEGAL OPERATOR IN CODE **') (SKIP(2),A);               00014700
```

```
     KO='0'B;                                                                  00014800
     RETURN;                                                                   00014900
/* EQUALITY OF TWO SERIES */                                                  00015000
EQL:  WS=UO(2)||DVRBL||TSS||ARG(2)||')';                                       00015100
     GO TO LS;                                                                 00015200
/* ADDITION OR SUBTRACTION OF TWO SERIES */                                   00015300
PMS: WS=UO(1)||DVRBL||TSS||ARG(1)||')'||CO||LP(2)||UO(2)||DVRBL||TSS||        00015400
      ARG(2)||')'||RP(2);                                                      00015500
     NADD=NADD+1;                                                              00015600
     GO TO LS;                                                                 00015700
/* MULTIPLICATION OF TWO SERIES */                                            00015800
MPY: IF ¬ZERO                                                                  00015900
     THEN DO; WS='SIGMA(IXV=1,ITS;'||DVRBL||'(IXV,'||ARG(1)||')*'||           00016000
               DVRBL||'(ITSP1-IXV,'||ARG(2)||'))';                             00016100
              NMTS=NMTS+1; NMTL=NMTL+1;                                        00016200
          END;                                                                 00016300
     ELSE                                                                      00016400
MDZ:      DO; WS=UO(1)||DVRBL||TSS||ARG(1)||')'||CO ||LP(2)||UO(2)||          00016500
               DVRBL||TSS||ARG(2)||')'||RP(2);                                 00016600
              NMUL=NMUL+1;                                                     00016700
          END;                                                                 00016800
     GO TO LS;                                                                 00016900
/* DIVISION OF ONE SERIES BY ANOTHER */                                       00017000
DVD: IF ZERO THEN GO TO MDZ;                                                   00017100
     WS='('||DVRBL||'(ITS,'||ARG(1)||                                          00017200
      ')-SIGMA(IXV=2,ITS;'||DVRBL||'(IXV,'||ARG(2)||')*'||                     00017300
      DVRBL||'(ITSP1-IXV,'||LHSARG||')))/'||DVRBL||'(1,'||ARG(2)||')';         00017400
     NMTS=NMTS+1; NMTL=NMTL-1;                                                 00017500
     GO TO LS;                                                                 00017600
INT: IF ZERO THEN GO TO LT;                                                    00017700
     IF SUBSTR(R(KCMAX,3),1,1)='#' THEN  GO TO LT;                             00017800
     WS=UO(2)||DVRBL||'(ITSM1,'||ARG(2)||')/FITSM1';                           00017900
     CALL CODE(LHS||WS,'1'B,'0'B);                                             00018000
     GO TO LT;                                                                 00018100
/* SERIES RAISED TO A POWER */                                                00018200
EXP:IF KA=3                                                                    00018300
     THEN DO; IF ¬ZERO THEN GO TO LT;                                          00018400
              WS=LP(1)||UO(1)||ARG(1)||RP(1)||'**'||LP(2)||ARG(2)||           00018500
               RP(2);                                                          00018600
              NADD=NADD+200;                                                   00018700
              GO TO LS;                                                        00018800
          END;                                                                 00018900
     IF ZERO                                                                   00019000
     THEN DO; WS=LP(1)||UO(1)||DVRBL||TSS||ARG(1)||')'||RP(1)||               00019100
               '**'||LP(2)||ARG(2)||RP(2);                                     00019200
              NADD=NADD+200;                                                   00019300
          END;                                                                 00019400
     ELSE DO; IF IED=1 THEN CALL REPLACE(ARG(2),'D','E','0'B);                 00019500
              ADS=BFTC(ARG(2)+1,0,IED);                                        00019600
```

```
            IF IED=1 THEN CALL REPLACE(ADS,'E','D','0'B);                    00019700
            IF VERIFY(SUBSTR(ARG(2),1,1),'+-')=0                             00019800
            THEN ADS='('||ADS||')';                                          00019900
            WS='SIGMA(IXV=1,ITSM1:('||ARG(2)||'-(IXV-1)*'||ADS||             00020000
              '/FITSM1)*'||DVRBL||'(IXV,'||LHSARG||')*'||DVRBL||             00020100
              '(ITSP1-IXV,'||ARG(1)||'))/'||DVRBL||'(1,'||ARG(1)||')';       00020200
            NATS=NATS+3; NATL=NATL-3; NMTS=NMTS+3; NMTL=NMTL-3;              00020300
         END;                                                                00020400
LS:  CALL CODE(LHS||WS,'1'B,'0'B);                                           00020500
LT:  DR(KCMAX)='1'B;                                                         00020600
     DO I=1 TO DMAX;                                                         00020700
        IF ¬DR(I) & DT(I,KCMAX)>=0 THEN DT(I,KCMAX)=DT(I,KCMAX)-1;           00020800
     END;                                                                    00020900
LU:  DO I=1 TO DMAX;                                                         00021000
        IF ¬DR(I) THEN GO TO LG;                                             00021100
     END;                                                                    00021200
     ZERO='0'B;                                                              00021300
     DR='0'B; DT=0;                                                          00021400
     GO TO LF;                                                               00021500
END CODE;                                                                    00021600
```

```
* PROCESS;                                                                00000100
 COL4: PROC(CS,RW,RMAX) RETURNS(BIN FIXED);                              00000200
 /*************************************************************************00000300
  * THE PROCEDURE SEARCHES COLUMN 4 OF THE RECURRENCE MATRIX TO FIND *    00000400
  * THE ROW NUMBER WHICH CONTAINS CS                                 *    00000500
  *************************************************************************/00000600
      DCL CS CHAR(*) VAR,I,RW(*,*) CHAR(*) VAR,RMAX BIN FIXED,           00000700
       CT CHAR(15) VAR;                                                  00000800
      CT=SUBSTR(CS,2-VERIFY(SUBSTR(CS,1,1),'+-'));                       00000900
      DO I=1 TO RMAX;                                                    00001000
         IF RW(I,4)=CT THEN RETURN(I);                                   00001100
      END;                                                               00001200
      PUT EDIT('*** ERROR IN INPUT EQUATIONS ***'||CS||'*** CAN NOT BE '||00001300
       'FOUND IN COLUMN 4 OF THE RECURRENCE MATRIX ***') (SKIP(2),A);    00001400
      STOP;                                                              00001500
 END COL4;                                                               00001600
```

```
* PROCESS;                                                                00000100
DMAT: PROC(RM,RMAX,DM);                                                   00000200
/*************************************************************************00000300
 * PROCEDURE CONSTRUCTS THE MATRIX D WHICH IS USED TO DETERMINE          *00000400
 * THE ORDER OF TAYLOR COEFFICIENT EVALUATION                            *00000500
 *************************************************************************/00000600
     DCL                                                                  00000700
      BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                        00000800
      COL4 ENTRY RETURNS(BIN FIXED),                                      00000900
      SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);        00001000
     DCL                                                                  00001100
      RM(*,*) CHAR( *) VAR,RMAX BIN FIXED,DM(*,*) BIN FIXED,              00001200
      R BIN FIXED,L(500,2) BIN FIXED,CH CHAR(1),(IVRBL,DVRBL)             00001300
      CHAR(4) VAR EXT,JCH CHAR(2) VAR,VN CHAR(4) VAR,DEBUG BIT(1) EXT;    00001400
/*                                                                       */00001500
     DM=-1;                                                               00001600
     DO R=1 TO RMAX;                                                      00001700
        IF DEBUG                                                          00001800
        THEN DO; IF R=1                                                   00001900
                 THEN PUT EDIT('DMAT ENTRY  ',R) (SKIP(2),A,F(3));        00002000
                 ELSE PUT EDIT(',',R) (A(1),F(3));                        00002100
             END;                                                         00002200
        N=0; I=1;                                                         00002300
        L(1,1)=R; L(1,2)=1+(RM(L(1,1),1)='='|RM(L(1,1),1)='%'             00002400
         |RM(L(1,1),1)='$');                                              00002500
LA:     IF RM(L(I,1),1)='''' THEN N=N+1;                                  00002600
        L1=L(I,1); L2=L(I,2)+1;                                           00002700
        CH=SUBSTR(RM(L1,L2),1,1);                                         00002800
        IF CH='?'                                                         00002900
        THEN DO; L(I+1,1)=SUBSTR(RM(L1,L2),2);                            00003000
                 IC4=COL4(RM(L1,L2),RM,RMAX);                             00003100
LC:              IF DM(R,IC4)<N+1 THEN DM(R,IC4)=N+1;                     00003200
                 L(I+1,2)=1+(RM(L(I+1,1),1)='='|                          00003300
                  RM(L(I+1,1),1)='%'|RM(L(I+1,1),1)='$');                 00003400
                 I=I+1;                                                   00003500
                 GO TO LA;                                                00003600
             END;                                                         00003700
        CALL SPAN(')'||RM(L1,L2),')',',(',VN);                            00003800
        IF VN¬='0'                                                        00003900
        THEN IF VERIFY(SUBSTR(VN,1,1),'+-')=0 THEN VN=SUBSTR(VN,2);       00004000
        IF VN=DVRBL                                                       00004100
        THEN DO; IC4=COL4(RM(L1,L2),RM,RMAX);                             00004200
                 IF DM(R,IC4)<N THEN DM(R,IC4)=N;                         00004300
                 GO TO LB;                                                00004400
             END;                                                         00004500
        IF CH='#' THEN GO TO LB;                                          00004600
        IF RM(L1,L2)=IVREL                                                00004700
        THEN DO; IF DM(R,1)<N THEN DM(R,1)=N;                             00004800
                 GO TO LB;                                                00004900
```

```
            END;                                                        00005000
        ELSE DO; IC4=COL4(RM(L1,L2),RM,RMAX);                           00005100
                 L(I+1,1)=IC4;                                          00005200
                 GO TO LC;                                              00005300
            END;                                                        00005400
LB:     IF L(I,2)=2 | RM(R,1)='=' | RM(R,1)='%'                         00005500
        THEN DO; I=I-1;                                                 00005600
                 IF I=0 THEN GO TO LR; ELSE GO TO LB;                   00005700
            END;                                                        00005800
        L(I,2)=2;                                                       00005900
        GO TO LA;                                                       00006000
LR:   END;                                                              00006100
ND:   END DMAT;                                                         00006200
```

```
* PROCESS;                                                                    00000100
 EXTRACT: PROC(STRING,WORD,EXS);                                             00000200
 /* PROCEDURE EXTRACTS THE SUMMATION OPERATOR FROM THE INPUT STRING    */00000300
      DCL                                                                    00000400
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                          00000500
       BLNCD ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),                          00000600
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1));            00000700
      DCL (STRING,WORD) CHAR(*) VAR,EXS CHAR(*) VAR,                         00000800
       (NSGMA,NSMTR) STATIC BIN FIXED EXT,EIS(3) CHAR(8) VAR EXT;            00000900
 /*                                                                    */00001000
      EXS='';                                                                00001100
      CALL EXT(WORD,MRKRA,MRKRB);                                            00001200
      IF MRKRA=0 THEN RETURN;                                                00001300
      DO I=1 TO 3;                                                           00001400
         IF EIS(I)=WORD THEN GO TO LA;                                       00001500
         CALL EXT(WORD,MI,LI);                                               00001600
         IF MI<MRKRA THEN RETURN;                                            00001700
 LA:  END;                                                                   00001800
      EXS=SUBSTR(STRING,MRKRA,MRKRB-MRKRA+1);                                00001900
      IF WORD=EIS(3)                                                         00002000
      THEN CALL REPLACE(STRING,EXS,'SGMA'||BFDC(NSGMA+1),'0'B);              00002100
      IF WORD=EIS(2)                                                         00002200
      THEN CALL REPLACE(STRING,EXS,'SMTR'||BFDC(NSMTR+1),'0'B);              00002300
      IF WORD=EIS(1) THEN CALL REPLACE(STRING,EXS||':','','0'B);             00002400
      EXS=SUBSTR(EXS,INDEX(EXS,'(')+1);                                      00002500
      EXS=SUBSTR(EXS,1,LENGTH(EXS)-1)||':';                                  00002600
 /*                                                                    */00002700
 EXT: PROC(W,MR,IL);                                                         00002800
      DCL W CHAR(*) VAR,MR,IL;                                               00002900
      MR,IL=0;                                                               00003000
 LD:  IF MR+1<LENGTH(STRING)                                                 00003100
      THEN IX=INDEX(SUBSTR(STRING,MR+1),W); ELSE IX=0;                       00003200
      IF IX=0                                                                00003300
      THEN                                                                   00003400
 LDA:      DO; MR=0; RETURN; END;                                            00003500
      ELSE MR=IX+MR;                                                         00003600
      IL=MR-1;                                                               00003700
 LE:  IX=INDEX(SUBSTR(STRING,IL+1),')');                                     00003800
      IF IX=0 THEN GO TO LDA; ELSE IL=IL+IX;                                 00003900
      IF BLNCD(SUBSTR(STRING,MR,IL-MR+1))¬=0 THEN GO TO LE;                  00004000
      IF INDEX(SUBSTR(STRING,MR,IL),'=')=0 THEN GO TO LD;                    00004100
      IF MR=1 | VERIFY(SUBSTR(STRING,MR-1,1),'+-*/)(=')=0 THEN RETURN;       00004200
      GO TO LD;                                                              00004300
 END EXT;                                                                    00004400
 END EXTRACT;                                                                00004500
```

```
* PROCESS;                                                                  00000100
 FUDGE: PROC(CTBL,DR,FTBL,R,RMAX,DT) RETURNS(BIT(1));                      00000200
 /*****************************************************************       00000300
  * PROCEDURE CONSTRUCTS THE RECURSIVE TAYLOR COEFFICIENTS FOR        *     00000400
  * FUNCTIONS SUCH AS EXP, SIN, COS, TAN ETC WHICH MAY APPEAR         *     00000500
  * IN THE DIFFERENTIAL EQUATIONS                                     *     00000600
  *****************************************************************/       00000700
      DCL                                                                   00000800
       CODE ENTRY(CHAR(*) VAR,BIT(1),BIT(1)),                               00000900
       COL4 ENTRY RETURNS(BIN FIXED);                                       00001000
      DCL                                                                   00001100
       DR(*) BIT(1),CTBL(*) CHAR(*) VAR,R(*,*) CHAR(*) VAR,FTBL(*,*)        00001200
       BIN FIXED,FN(13,2) CHAR(6) VAR EXT,RMAX BIN FIXED,                   00001300
       (ARG(2) CHAR(25) VAR, (CA3,ZERO) BIT(1), (LHS,LHSARG) CHAR(15)       00001400
       VAR, (LP(2),RP(2),UO(2)) CHAR(1) VAR, KCMAX, KFMAX, TSS CHAR(5)      00001500
       VAR, (IVRBL,DVRBL) CHAR(4) VAR  ) EXT,VNLHS CHAR(15) VAR,            00001600
       LHSA(3) CHAR(15) VAR, FNL( 9) LABEL INIT(EXP,L10,LN,                 00001700
       SCT,SCT,SCT,HSCT,HSCT,HSCT),PM CHAR(1) VAR,                          00001800
       (NMUL,NADD,NMTS,NMTL,NATS,NATL) EXT,IED EXT,DT(*,*) BIN FIXED,       00001900
       ALPHA(2) CHAR(29) VAR INIT('4.342945E-1','4.34294481903251BD-1');    00002000
 /*                                                                         */ 00002100
      DO KF=1 TO KFMAX;                                                     00002200
         IF KCMAX<=FTBL(KF,2) & KCMAX>=FTBL(KF,1) THEN GO TO LA;            00002300
      END;                                                                  00002400
      PUT EDIT('** FUNCTION NUMBER ',KCMAX,' IS NOT IN TABLE **')           00002500
       (SKIP(2),A,F(2),A);                                                  00002600
      RETURN('0'B);                                                         00002700
 LA:  I=0;                                                                  00002800
      DO K=FTBL(KF,1) TO FTBL(KF,2);                                        00002900
         DR(K)='1'B;                                                        00003000
         DO J=1 TO RMAX;                                                    00003100
            IF ¬DR(J) & DT(J,K)>=0 THEN DT(J,K)=DT(J,K)-1;                  00003200
         END;                                                               00003300
         I=I+1;                                                             00003400
         LHSA(I)=CTBL(COL4(R(K,4),R,RMAX));                                 00003500
      END;                                                                  00003600
      NF=R(KCMAX,2);                                                        00003700
      IF ¬ZERO THEN GO TO LB;                                               00003800
      I=0;                                                                  00003900
      DO K=FTBL(KF,1) TO FTBL(KF,2);                                        00004000
         I=I+1;                                                             00004100
         KX=R(K,2);                                                         00004200
         VNLHS=DVRBL||'(1,'||LHSA(I)||')=';                                 00004300
         IF CA3                                                             00004400
         THEN CALL CODE(VNLHS||FN(KX,2)||'('||UO(2)||ARG(2)||'))',"0'B,     00004500
                   '0'B);                                                   00004600
         ELSE CALL CODE(VNLHS||FN(KX,2)||'('||UO(2)||DVRBL||'(1,'||         00004700
                   ARG(2)||'))','0'B,'0'B);                                 00004800
      END;                                                                  00004900
```

```
     GO TO LZ;                                                          00005000
LB:  IF CA3 THEN RETURN('1'B); ELSE GO TO FNL(NF);                      00005100
/* SERIES COEFFICIENTS FOR EXPONENTIAL FUNCTION */                      00005200
EXP: CALL CODE(LHS||'SIGMA(IXV=2,ITS:'||'(IXV-1)*'||                    00005300
      DVRBL||'IXV,'||ARG(2)||')*'||DVRBL||'(ITSP1-IXV,'||               00005400
      LHSA(1)||')/FITSM1', '1'B,'0'B);                                  00005500
     NMUL=NMUL+1; NADD=NADD+2; NMTL=NMTL-1;                             00005600
     GO TO LZ;                                                          00005700
/* SERIES COEFFICIENTS FOR LOG BASE 10 FUNCTION */                      00005800
L10: CALL CODE('IF (ITS.EQ.2) '||LHS||ALPHA(IED+1)||'*'||DVRBL ||'(2,' 00005900
      ||ARG(2)||')/'||DVRBL||'(1,'||ARG(2)||',','''B,'0'B);             00006000
     CALL CODE('IF (ITS.GT.2) '||LHS||'('||ALPHA(IED+1)||'*'            00006100
      ||DVRBL||TSS||ARG(2)||')-SIGMA(IXV=2,ITSM1;(IXV-1)*'||            00006200
      DVRBL||'(ITSP1-IXV,'||ARG(2)||')*'||DVRBL||'(IXV,'||              00006300
      LHSA(1)||'))/FITSM1)/' ||DVRBL||'(1,'||ARG(2)||')',1'B,'0'B);    00006400
     NMUL=NMUL+3; NADD=NADD+1; NMTS=NMTS+2; NMTL=NMTL-2;                00006500
     GO TO LZ;                                                          00006600
/* SERIES COEFFICIENTS FOR LOG BASE E FUNCTION */                       00006700
LN:  CALL CODE('IF (ITS.EQ.2) '||LHS||DVRBL||TSS||ARG(2)||')/'          00006800
      ||DVRBL||'(1,'||ARG(2)||')',0'B,'0'B);                            00006900
     CALL CODE('IF (ITS.GT.2) '||LHS||'('||DVRBL||TSS||ARG(2)           00007000
      ||')-'||'SIGMA(IXV=2,ITSM1;(IXV-1)*'||DVRBL||'(ITSP1-IXV,'||      00007100
      ARG(2)||')*'||DVRBL||'(IXV,'||LHSARG||'))/FITSM1/'||DVRBL||'(1,' 00007200
      ||ARG(2)||')/'||DVRBL||'(1,'||ARG(2)||')',1'B,'0'B);              00007300
     NMUL=NMUL+2; NADD=NADD+1; NMTS=NMTS+2; NMTL=NMTL-2;                00007400
     GO TO LZ;                                                          00007500
/* SERIES COEFFICIENTS FOR THE SIN, COS, TAN FUNCTIONS */               00007600
SCT: PM='-';                                                            00007700
     GO TO LC;                                                          00007800
/* SERIES COEFFICIENTS FOR THE HYPERBOLIC SINH, COSH, TANH FUNCTIONS */ 00007900
HSCT: PM='';                                                            00008000
LC: CALL CODE(DVRBL||TSS||LHSA(1)||')=SIGMA(IXV=2,ITS:'(IXV-1)*'||      00008100
      DVRBL||'(IXV,'||ARG(2)||')*'||DVRBL||'(ITSP1-IXV,'||LHSA(2)       00008200
      ||'))/FITSM1' ,'1'B,'0'B);                                        00008300
     CALL CODE(DVRBL||TSS||LHSA(2)||')='||PM||'SIGMA(IXV=2,ITS:'||      00008400
      '(IXV-1)*'||DVRBL||'IXV,'||ARG(2)||')*'||DVRBL||'(ITSP1-IXV,'     00008500
      ||LHSA(1)||'))/FITSM1' ,'1'B,'0'B);                               00008600
     CALL CODE(DVRBL||TSS||LHSA(3)||')='||DVRBL||TSS||LHSA(1)||         00008700
      ')-SIGMA(IXV=2,ITS:'||DVRBL||'(IXV,'||LHSA(2)||')*'||DVRBL||      00008800
      '(ITSP1-IXV,'||LHSA(3)||')))/'||DVRBL||'(1,'||LHSA(2)||')',      00008900
      '1'B,'0'B);                                                       00009000
     NMUL=NMUL+3; NADD=NADD+1; NMTS=NMTS+6; NMTL=NMTL-6;                00009100
LZ: RETURN('1'B);                                                       00009200
END FUDGE;                                                              00009300
```

```
* PROCESS;                                                                  00000100
 INPUT: PROC(ERROR,CC);                                                    00000200
 /*************************************************************************  00000300
  * PROCEDURE READS THE DEFINING SYSTEM OF DIFFERENTIAL EQUATIONS     *     00000400
  * FROM THE SYSIN DATA SET, AND CHECKS TO SEE THAT THE EQUATIONS     *     00000500
  * ARE BALANCED WITH RESPECT TO PARENTHESES                          *     00000600
  *************************************************************************/ 00000700
      DCL                                                                   00000800
       BLNCD ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),                         00000900
       COUNT ENTRY(CHAR(*) VAR,CHAR(1)) RETURNS(BIN FIXED),                 00001000
       DELETE ENTRY(CHAR(*) VAR,CHAR(1)),                                   00001100
       SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);         00001200
      DCL                                                                   00001300
       (ERROR,CC) BIN FIXED, LINE CHAR(80) VAR,                             00001400
       CW(2) CHAR(25) VAR INIT('DIFFERENTIALEQUATIONS',                     00001500
       'INITIALVALUES'),  CS CHAR(400) VAR EXT, WS CHAR(400) VAR EXT,       00001600
       IED EXT, C48 CHAR(1) EXT,(IVRBL,DVRBL) CHAR(4) VAR EXT;              00001700
 /*                                                                       */ 00001800
      ON ENDFILE(SYSIN)                                                     00001900
      BEGIN; PUT EDIT('** EOF READING SYSIN **') (SKIP(2),A);               00002000
             GO TO LPA;                                                     00002100
      END;                                                                  00002200
      IF CC=1                                                               00002300
      THEN DO; MRKR=0;                                                      00002400
               PUT EDIT('** TAYLOR SERIES PROGRAM JAN. 1973'||              00002500
                ' VERSION - LISTING OF INPUT EQUATIONS **') (A);            00002600
               ERROR=1;                                                     00002700
           END;                                                             00002800
      ELSE MRKR=INDEX(CS,'#');                                              00002900
      PUT SKIP;                                                             00003000
 LP:  GET EDIT(LINE) (A(80));                                               00003100
      PUT EDIT(LINE) (SKIP,COLUMN(4),A(80));                                00003200
      LINE=SUBSTR(LINE,1,72);                                               00003300
      CALL DELETE(LINE,' ');                                                00003400
      IF CC¬=0                                                              00003500
      THEN DO; IF INDEX(LINE,CW(CC))¬=0 THEN GO TO LQ;                      00003600
               PUT EDIT('** THE FOLLOWING CONTROL CARD IS INVALID **',      00003700
                LINE) (SKIP(2),A,SKIP,A);                                   00003800
 LPA:          ERROR=3;                                                     00003900
               RETURN;                                                      00004000
 LQ:           IF CC=1                                                      00004100
               THEN DO; IF INDEX(LINE,'DP')¬=0                              00004200
                        THEN DO; IED=1; C48='8'; END;                       00004300
                        ELSE DO; IED=0; C48='4'; END;                       00004400
                        IF INDEX(LINE,'(')¬=0                               00004500
                        THEN DO; CALL SPAN(LINE,'(',',',IVRBL);             00004600
                                 CALL SPAN(LINE,',',')',DVRBL);             00004700
                             END;                                           00004800
                    END;                                                    00004900
```

```
            CC=0; PUT SKIP; GO TO LP;                                    00005000
         END;                                                            00005100
      CS=CS!!LINE;                                                       00005200
      IF INDEX(LINE,':')=0  THEN GO TO LP;                               00005300
      PUT SKIP;                                                          00005400
      CS=SUBSTR(CS,1,LENGTH(CS)-1)||';';                                 00005500
      DO WHILE(MRKR<LENGTH(CS));                                         00005600
         MC=INDEX(SUBSTR(CS,MRKR+1),';');                                00005700
         WS=SUBSTR(CS,MRKR+1,MC-1);                                      00005800
         MRKR=MRKR+MC;                                                   00005900
         IF BLNCD(WS)¬=0                                                 00006000
         THEN DO; PUT EDIT('** THE FOLLOWING EXPRESSION HAS AN '||       00006100
                   'INCORRECT PAIRING OF PARENTHESES **',WS)             00006200
                   (SKIP(2),A,SKIP,A);                                   00006300
                  ERROR=2;                                               00006400
              END;                                                       00006500
         IF COUNT(WS,'=')¬=1                                             00006600
         THEN DO; PUT EDIT('** SYNTAX ERROR IN THE FOLLOWING EXPRESSION'00006700
                   ||' **',WS) (SKIP(2),A,SKIP,A);                       00006800
                  ERROR=2;                                               00006900
              END;                                                       00007000
      END;                                                               00007100
ND: END INPUT;                                                           00007200
```

```
* PROCESS;                                                                00000100
 OP: PROC(CH) RETURNS(BIN FIXED);                                        00000200
 /*************************************************************************00000300
  * PROCEDURE COMPARES THE INPUT CHARACTER CH WITH THE CHARACTERS     *   00000400
  * +, -, *, /, =, (, ), #, $, %                                     *   00000500
  *************************************************************************/00000600
      DCL CH CHAR(1),I,OC(10) CHAR(1) EXT;                               00000700
      DO I=1 TO 10;                                                      00000800
         IF CH=OC(I) THEN RETURN(I);                                     00000900
      END;                                                               00001000
      RETURN(0);                                                         00001100
 END OP;                                                                 00001200
```

```
* PROCESS;                                                                00000100
 OPTMZE: PROC(M,RMAX);                                                   00000200
 /* PROCEDURE ELIMINATES REDUNDANT OPERATIONS FROM THE R MATRIX */       00000300
      DCL                                                                00000400
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                      00000500
       (I,J,K,R,RMAX) BIN FIXED,IMP BIT(1),M(*,*) CHAR(15) VAR,          00000600
       IX(2,2) INIT(2,3,3,2);                                            00000700
 LA:  R=0; IMP='0'B;                                                     00000800
 LAB: R=R+1; I=R+1;                                                      00000900
 LB:  DO WHILE(I<=RMAX);                                                 00001000
         IF M(R,1)¬=M(I,1) THEN GO TO LD;                                00001100
         IF M(R,1)='+'|M(R,1)='-'|M(R,1)='*'|M(R,1)='/'                  00001200
         THEN JMAX=2; ELSE JMAX=1;                                       00001300
         DO J=1 TO JMAX;                                                 00001400
            DO K=1 TO 2;                                                 00001500
               IF M(R,K+1)¬=M(I,IX(K,J)) THEN GO TO LC;                  00001600
            END;                                                         00001700
            IMP='1'B;                                                    00001800
            CALL RAD(I,R);                                               00001900
            GO TO LE;                                                    00002000
 LC:     END;                                                            00002100
 LD:     I=I+1;                                                          00002200
 LE:  END;                                                               00002300
      IF R<RMAX-1 THEN GO TO LAB;                                        00002400
      IF IMP THEN GO TO LA;                                              00002500
 /*                                                                      */00002600
 RAD: PROC(I,J);                                                         00002700
 /************************************************************************ 00002800
  * PROCEDURE DELETES ROW I, AND REPLACES REFERENCES TO ROW I WITH      * 00002900
  * ROW J IN THE RECURRENCE MATRIX                                      * 00003000
  ************************************************************************/00003100
      DCL I,J,K,L,ROW BIN FIXED;                                         00003200
      K=I+1;                                                             00003300
      DO WHILE(K<=RMAX);                                                 00003400
         M(K-1,1)=M(K,1);                                                00003500
         DO L=2 TO 4;                                                    00003600
            IF SUBSTR(M(K,L),1,1)¬='?'                                   00003700
            THEN DO; M(K-1,L)=M(K,L); GO TO LF; END;                     00003800
            ROW=SUBSTR(M(K,L),2);                                        00003900
            IF ROW=I THEN ROW=J; ELSE IF ROW>I THEN ROW=ROW-1;           00004000
            M(K-1,L)='?' || BFDC(ROW);                                   00004100
 LF:     END;                                                            00004200
         K=K+1;                                                          00004300
      END;                                                               00004400
      RMAX=RMAX-1;                                                       00004500
 END RAD;                                                                00004600
 END OPTMZE;                                                             00004700
```

```
* PROCESS;                                                                00000100
 RMAT: PROC(CS,R,RMAX,KO) RECURSIVE;                                      00000200
 /*****************************************************************     00000300
  * FACTORIZATION OF THE DIFFERENTIAL SYSTEM INTO CANONICAL FORM      *   00000400
  * USING THE ALGORITHM DESCRIBED IN 'BOUNDED CONTEXT TRANSLATION'.   *   00000500
  * BY R. GRAHAM, AFIPS-SJCC V 25 (1964), P. 21                       *   00000600
  *****************************************************************/     00000700
      DCL                                                                 00000800
       OP ENTRY(CHAR(1)) RETURNS(BIN FIXED),                              00000900
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                       00001000
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1)),         00001100
       SFNL ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),                        00001200
       SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);       00001300
      DCL P(11) BIN FIXED STATIC          INIT(6,6,7,7,5,3,4,1,2,8,8),    00001400
       S( 500) CHAR(15) VAR INIT(( 500) ''), L( 250) CHAR(15) VAR,        00001500
       TYPE BIN FIXED,R(*,*) CHAR(15) VAR,CS CHAR(*) VAR,                 00001600
       WT CHAR(400) VAR,FN CHAR(4) VAR,FTBL(100,2) BIN FIXED EXT,         00001700
       WS CHAR(400) VAR,RMAX BIN FIXED,DVRBL CHAR(4) VAR EXT;             00001800
      DCL CST(100) CHAR(25) VAR EXT,IC EXT,KO BIT(1),IED EXT,             00001900
       NEST BIN FIXED STATIC INIT(0),DEBUG BIT(1) EXT,NEQ EXT,            00002000
       PMD CHAR(36) VAR EXT;                                              00002100
 /*                                                                       */ 00002200
      KO='1'B;                                                            00002300
      NEST=NEST+1;                                                        00002400
      IF NEST=1 & NEQ=1 & DEBUG                                           00002500
      THEN DO;                                                            00002600
                PUT PAGE;                                                 00002700
                PUT EDIT('RMAT ENTRY',' LEVEL K TYPE (C, U, V | E)',      00002800
                 ' R OP',' A(1)',' A(2)',' A(3)') (SKIP(2),A,SKIP(1),A    00002900
                 X(33),A,X(5),A,3 (X(11),A));                             00003000
      END;                                                                00003100
      DO I=1 TO LENGTH(CS)/45+1;                                          00003200
         PUT EDIT(SUBSTR(CS,1+(I-1)*45,MIN(45,LENGTH(CS)-(I-1)*45)))      00003300
          (SKIP,X(15),A);                                                 00003400
      END;                                                                00003500
      J=1;                                                                00003600
      K=2;                                                                00003700
      MRKR=2;                                                             00003800
      S(1)=SUBSTR(CS,1,1);                                                00003900
      TYPE=8;                                                             00004000
      L(1)=S(1);                                                          00004100
 L1:  IF S(K)='' THEN CALL CHECK;                                         00004200
      IF DEBUG THEN PUT EDIT(NEST,K,TYPE,S(K)) (SKIP,3 F(4),X(3),A);      00004300
      IF TYPE<1 THEN GO TO L2;                                            00004400
      IF TYPE¬=6 THEN GO TO LA;                                           00004500
 L2:  J=J+1;                                                              00004600
      L(J)=S(K);                                                          00004700
 L3:  K=K+1;                                                              00004800
      IF K>DIM(S,1)                                                       00004900
```

```
     THEN DO; PUT EDIT('** OVERFLOW IN S TABLE **') (SKIP(2),A);          00005000
          KO='0'B; RETURN;                                                00005100
        END;                                                              00005200
     GO TO L1;                                                            00005300
LA:  IF P(OP(L(J-1)))<P(TYPE) THEN GO TO LB;                              00005400
     RMAX=RMAX+1;                                                         00005500
     IF L(J-1)¬='=' THEN GO TO LAB;                                       00005600
     IF LENGTH(L(J-2))<LENGTH(DVRBL) | SUBSTR(L(J-2),1,LENGTH(DVRBL))     00005700
      ¬=DVRBL THEN GO TO LAA;                                             00005800
     CALL SPAN(L(J-2),'0','9')',FN);                                      00005900
     R(RMAX,1)='S';                                                       00006000
     R(RMAX,2)=DVRBL||'('||FN||')';                                       00006100
     R(RMAX,3)=L(J);                                                      00006200
     R(RMAX,4)=R(RMAX,2);                                                 00006300
     GO TO LAC;                                                           00006400
LAA: IF SUBSTR(L(J),1,1)¬='?' | SUBSTR(L(J),2)¬=BFDC(RMAX-1)              00006500
      THEN GO TO LAB;                                                     00006600
     RMAX=RMAX-1;                                                         00006700
     R(RMAX,4)=L(J-2);                                                    00006800
     GO TO LAC;                                                           00006900
LAB: IF L(J-1)='**' & INDEX(L(J),'#')¬=0                                  00007000
     THEN DO; IF CST(SUBSTR(L(J),2))¬='2' THEN GO TO LAZ;                 00007100
          R(RMAX,1)='*';                                                  00007200
          R(RMAX,2)=L(J-2);                                               00007300
          R(RMAX,3)=L(J-2);                                               00007400
        END;                                                              00007500
     ELSE                                                                 00007600
LAZ:      DO; R(RMAX,1)=L(J-1);                                           00007700
          R(RMAX,2)=L(J-2);                                               00007800
          R(RMAX,3)=L(J);                                                 00007900
        END;                                                              00008000
     IF L(J-1)='='                                                        00008100
     THEN R(RMAX,4)=R(RMAX,2); ELSE R(RMAX,4)='?'||BFDC(RMAX);            00008200
LAC: IF DEBUG THEN PUT EDIT(RMAX,(R(RMAX,M) DO M=1 TO 4))                 00008300
      (SKIP,X(60),F(2),X(1),A(2),X(5),4 A(15));                           00008400
     J=J-2;                                                               00008500
     L(J)='?' || BFDC(RMAX);                                              00008600
     GO TO LA;                                                            00008700
LB:  IF TYPE=7 THEN DO; L(J-1)=L(J); J=J-1; GO TO L3; END;                00008800
     IF TYPE¬=9 THEN GO TO L2;                                            00008900
     IF K=2 THEN CS=S(2); ELSE CS='?'||BFDC(RMAX);                        00009000
     NEST=NEST-1;                                                         00009100
/*                                                                        */ 00009200
CHECK: PROC RECURSIVE;                                                    00009300
/************************************************************************ 00009400
 * PROCEDURE SCANS A SEQUENCE OF CHARACTERS BEGINNING WITH MRKR TO      * 00009500
 * DETERMINE WHETHER THEY SPECIFY A CONSTANT, VARIABLE, OR OPERATOR     * 00009600
 ************************************************************************/ 00009700
     DCL                                                                  00009800
```

```
      BLNCD ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),                    00009900
      FLIP BIT(1), CH CHAR(1),STRING CHAR(15) VAR INIT(''),           00010000
      BREAKB ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR),              00010100
      BREAKF ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);              00010200
/*                                                                    */ 00010300
     CH=SUBSTR(CS,MRKR,1);                                            00010400
     S(K)=CH;                                                         00010500
     MRKR=MRKR+1;                                                     00010600
     IO=OP(CH);                                                       00010700
     IF IO=0 THEN GO TO LG;                                           00010800
     IF (TYPE=5|TYPE=6)&IO<3                                          00010900
     THEN IF VERIFY(SUBSTR(CS,MRKR,1),'.0123456789')¬=0               00011000
          THEN GO TO LGA; ELSE GO TO LJ;                              00011100
     IF TYPE>0&TYPE¬=6&TYPE¬=7&IO¬=6&IO¬=7                             00011200
     THEN GO TO LJ;                                                   00011300
     IF IO=9 THEN DO; TYPE=IO; GO TO ND; END;                         00011400
     CH=SUBSTR(CS,MRKR,1);                                            00011500
     NOP=OP(CH);                                                      00011600
     IF NOP=3 & IO=3 THEN DO; S(K)=S(K) || CH; MRKR=MRKR+1; END;      00011700
     IF LENGTH(S(K))=2 THEN TYPE   =11; ELSE TYPE   =IO;              00011800
     GO TO ND;                                                        00011900
LG:  IF VERIFY(S(K),'ABCDEFGHIJKLMNOPQRSTUVWXYZ?')¬=0 THEN GO TO LJ;  00012000
/* PROCESS A VARIABLE */                                             00012100
LGA: FLIP='0'B; TYPE=0; WS=S(K);                                      00012200
LH:  CH=SUBSTR(CS,MRKR,1);                                            00012300
     NOP=OP(CH);                                                      00012400
     IF NOP=0 THEN GO TO LI;                                          00012500
     IF NOP¬=6 & NOP¬=7 & ¬FLIP THEN GO TO LIA;                       00012600
     IF BLNCD(WS||CH)<0 THEN GO TO LIA;                               00012700
     FLIP='1'B;                                                       00012800
LI:  WS=WS||CH;                                                       00012900
     MRKR=MRKR+1;                                                     00013000
     IF ¬FLIP | BLNCD(WS)¬=0 THEN GO TO LH;                           00013100
     CALL SPAN('')'||WS,'.','(',WT);                                  00013200
     IF WT=DVRBL THEN                                                 00013300
LIA:  DO; S(K)=WS; GO TO ND; END;                                     00013400
     LFN=SFNL(WS);                                                    00013500
     IF LFN=0 THEN GO TO LIA;                                         00013600
     CALL BREAKF(WS,'(',WT);                                          00013700
     CALL BREAKB(WS,')',WT);                                          00013800
     IF VERIFY(WS,PMD) =0 THEN GO TO LIB;                             00013900
     WS='#TEMPN='||WS||'$';                                           00014000
     CALL RMAT(WS,R,RMAX,KO);                                         00014100
     IF¬KO THEN RETURN;                                               00014200
     R(RMAX,4)='?'||BFDC(RMAX);                                       00014300
LIB: CALL SFNC(LFN,WS,R,RMAX,S(K),FTBL,KO);                           00014400
     IF ¬KO THEN RETURN;                                              00014500
     GO TO ND;                                                        00014600
/* PROCESS A CONSTANT */                                             00014700
```

```
LJ:   TYPE=-1;                                                          00014800
      IF IC=100                                                         00014900
      THEN DO; PUT EDIT('** OVERFLOW IN CONSTANT TABLE **') (SKIP(2),A);00015000
               KC='0'B; RETURN;                                         00015100
           END;                                                         00015200
      IC=IC+1;                                                          00015300
      CST(IC)=S(K);                                                     00015400
LJA: CH=SUBSTR(CS,MRKR,1);                                              00015500
      IF VERIFY(CH,'.0123456789ED+-')¬=0 THEN GO TO LK;                 00015600
      IF (CH='+'|CH='-')&VERIFY(SUBSTR(CST(IC),LENGTH(CST(IC)),1),      00015700
       'ED')¬=0 THEN GO TO LK;                                          00015800
      IF (CH='E' | CH='D') & VERIFY(STRING,'.1234567890')¬=0            00015900
       THEN GO TO LK;                                                   00016000
      CST(IC)=CST(IC)||CH;                                              00016100
      STRING=STRING||CH;                                                00016200
      MRKR=MRKR+1;                                                      00016300
      GO TO LJA;                                                        00016400
LK:   IF IC¬=1                                                          00016500
      THEN DO; IF IED=1 & INDEX(CST(IC),'.')¬=0                         00016600
               THEN DO; CALL REPLACE(CST(IC),'E','D','0'B);             00016700
                        IF INDEX(CST(IC),'D')=0                         00016800
                        THEN CST(IC)=CST(IC)||'D0';                     00016900
                    END;                                                00017000
               DO I=1 TO IC-1;                                          00017100
                  IF CST(I)¬=CST(IC) THEN GO TO LJB;                    00017200
                  S(K)='#'||BFDC(I);                                    00017300
                  IC=IC-1;                                              00017400
                  RETURN;                                               00017500
LJB:           END;                                                     00017600
           END;                                                         00017700
      S(K)='#' || BFDC(IC);                                             00017800
      RETURN;                                                           00017900
ND: END CHECK;                                                          00018000
END RMAT;                                                               00018100
```

```
* PROCESS;                                                                00000100
 SFNC: PROC(LFN,WS,R,RMAX,S,FTBL,KO);                                    00000200
 /*********************************.*********************************** 00000300
  * PROCEDURE INSERTS THE PROPER ENTRIES IN THE R MATRIX FOR AN       *  00000400
  * ALLOWABLE FUNCTION REFERENCE                                      *  00000500
  ********************************************************************/ 00000600
      DCL BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),RMAX BIN FIXED,    00000700
       R(*,*) CHAR( *) VAR,S CHAR(*) VAR,WS CHAR(*) VAR,                 00000800
       CLFN CHAR(15) VAR,FTBL(*,*) BIN FIXED,KFMAX EXT,DEBUG BIT(1) EXT.00000900
       KFL(10) INIT(1,2,3,4,4,4,7,7,7,10),                               00001000
       KFU(10) INIT(1,2,3,6,6,6,9,9,9,10);                               00001100
 /*                                                                      */ 00001200
      CLFN=BFDC(LFN);                                                    00001300
      DO I=1 TO RMAX;                                                    00001400
         IF  R(I,1)¬='%' |  R(I,2)¬=CLFN |  R(I,3)¬=WS                    00001500
          THEN GO TO LA;                                                 00001600
          ELSE DO; S='?' || BFDC(I); RETURN; END;                        00001700
 LA:  END;                                                               00001800
      DO K=KFL(LFN) TO KFU(LFN);                                         00001900
         RMAX=RMAX+1;                                                    00002000
         IF K=10                                                         00002100
         THEN DO; R(RMAX,1)='**';                                        00002200
                  R(RMAX,2)=WS;                                          00002300
                  R(RMAX,3)='#1';                                        00002400
                  GO TO LB;                                              00002500
              END;                                                       00002600
         R(RMAX,1)='%';                                                  00002700
         R(RMAX,2)=BFDC(K);                                              00002800
         R(RMAX,3)=WS;                                                   00002900
 LB:     R(RMAX,4)='?' || BFDC(RMAX);                                    00003000
         IF DEBUG                                                        00003100
         THEN PUT EDIT(RMAX,(R(RMAX,M) DO M=1 TO 4)) (SKIP,X(60),F(2),   00003200
               X(1),A(2),X(5),A A(15));                                  00003300
         IF K=LFN THEN S='?'||BFDC(RMAX);                                00003400
      END;                                                               00003500
      KFMAX=KFMAX+1;                                                     00003600
      IF KFMAX>DIM(FTBL,1)                                               00003700
      THEN DO; PUT EDIT('** OVERFLOW IN FUNCTION TABLE **')              00003800
                (SKIP(2),A);                                             00003900
               KO='0'B; RETURN;                                          00004000
           END;                                                          00004100
      FTBL(KFMAX,2)=RMAX;                                                00004200
      FTBL(KFMAX,1)=RMAX-KFU(LFN)+KFL(LFN);                              00004300
 END SFNC;                                                               00004400
```

```
* PROCESS;                                                                    00000100
 SFNL: PROC(WS) RETURNS(BIN FIXED);                                          00000200
 /* PROCEDURE DETERMINES WHETHER WS IS AN ALLOWABLE FUNCTION */             00000300
       DCL                                                                   00000400
        WS CHAR(*) VAR,NF EXT,F CHAR(6) VAR,FN(10,2) CHAR(6) VAR EXT;        00000500
       F=SUBSTR(WS,1,INDEX(WS,'(')-1);                                       00000600
       DO I=1 TO NF; DO J=1 TO 2;                                            00000700
          IF F=FN(I,J) THEN RETURN(I);                                       00000800
       END; END;                                                             00000900
       RETURN(0);                                                            00001000
 END SFNL;                                                                   00001100
```

```
* PROCESS;                                                                    00000100
 SIGMA: PROC(STRING) RECURSIVE;                                              00000200
 /* PROCEDURE CONSTRUCTS A FORTRAN DO LOOP FOR A SUMMATION OPERATOR  */      00000300
      DCL                                                                    00000400
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),                          00000500
       BREAKF ENTRY(CHAR(*) VAR,CHAR(1),CHAR(*) VAR),                        00000600
       CODE ENTRY(CHAR(*) VAR,BIT(1),BIT(1)),                                00000700
       EXTRACT ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);                   00000800
      DCL EXA CHAR(400) VAR,NSGMA BIN FIXED STATIC EXT,                      00000900
       (STN,SMN) CHAR(5) VAR,EIS(3) CHAR(8) VAR EXT,STRING CHAR(*) VAR;      00001000
 /*                                                                  */      00001100
      NSGMA=NSGMA+1;                                                         00001200
      SMN=BFDC(NSGMA);                                                       00001300
      STN=BFDC(1000+NSGMA);                                                  00001400
      CALL BREAKF(STRING,':',EXA);                                           00001500
      CALL CODE('SGMA'||SMN||'=0.0',''B,''B);                                00001600
      CALL CODE('DO '||STN||' '||EXA,''B,''B);                               00001700
      CALL CODE(STN||'SGMA'||SMN||'=SGMA'||SMN||'+('||SUBSTR(STRING,1,       00001800
       LENGTH(STRING)-1)||')',''B,''B);                                      00001900
 END SIGMA;                                                                  00002000
```

```
* PROCESS;                                                                    00000100
 STINT: PROC;                                                                00000200
 /* PROCEDURE INITIALIZES ALL EXTERNAL BIT AND CHARACTER STRING VBL'S */      00000300
      DCL(                                                                   00000400
       NF INIT(10),(IVRBL,DVRBL) CHAR(4) VAR,EIS(3) CHAR(8) VAR,             00000500
       OC(10) CHAR(1),FN(13,2) CHAR(6) VAR,SYSA BIT(1),PMD CHAR(36) VAR      00000600
       ) EXT;                                                                00000700
 /*                                                                          */00000800
      SYSA='1'B;                                                             00000900
      PMD='ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';                            00001000
      IVRBL='T'; DVRBL='Y';                                                  00001100
      EIS(1)='EQUATION'; EIS(2)='INTEGRAL'; EIS(3)='SIGMA';                  00001200
      OC( 1)='+'; OC( 2)='-'; OC( 3)='*'; OC( 4)='/'; OC( 5)='=';            00001300
      OC( 6)='('; OC( 7)=')'; OC( 8)='#'; OC( 9)='$'; OC(10)='%';            00001400
      FN( 1,1)=   'EXP'; FN( 1,2)=  'DEXP';                                  00001500
      FN( 2,1)='ALOG10'; FN( 2,2)='DLOG10';                                  00001600
      FN( 3,1)=  'ALOG'; FN( 3,2)=  'DLOG';                                  00001700
      FN( 4,1)=   'SIN'; FN( 4,2)=  'DSIN';                                  00001800
      FN( 5,1)=   'COS'; FN( 5,2)=  'DCOS';                                  00001900
      FN( 6,1)=   'TAN'; FN( 6,2)=  'DTAN';                                  00002000
      FN( 7,1)=  'SINH'; FN( 7,2)= 'DSINH';                                  00002100
      FN( 8,1)=  'COSH'; FN( 8,2)= 'DCOSH';                                  00002200
      FN( 9,1)=  'TANH'; FN( 9,2)= 'DTANH';                                  00002300
      FN(10,1)=  'SQRT'; FN(10,2)= 'DSQRT';                                  00002400
 END STINT;                                                                  00002500
```

```
* PROCESS;                                                              00000100
 BFDC: PROC(N) RETURNS(CHAR(15) VAR);                                   00000200
 /*************************************************************** *******  00000300
  * BIN FIXED NUMBER N IS CONVERTED TO CHARACTER STRING WITH ALL      *   00000400
  * BLANKS RESULTING FROM THE CONVERSION DELETED                      *   00000500
  ********************************************************************/  00000600
      DCL DELETE ENTRY(CHAR(*) VAR,CHAR(1)),CS CHAR(15) VAR;            00000700
      CS=CHAR(N);                                                       00000800
      CALL DELETE(CS,' ');                                              00000900
      RETURN(CS);                                                       00001000
 END BFDC;                                                              00001100
```

```
* PROCESS;                                                              00000100
 BFTC: PROC(BF,IED) RETURNS(CHAR(50) VAR);                              00000200
 /*************************************************************** *****  00000300
  * BIN FLOAT(53) NUMBER BF IS CONVERTED TO CHARACTER STRING WITH     *   00000400
  * ALL BLANKS RESULTING FROM THE CONVERSION DELETED                  *   00000500
  ********************************************************************/  00000600
      DCL                                                               00000700
       DELETE ENTRY(CHAR(*) VAR,CHAR(1)),                               00000800
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1)),       00000900
       BF BIN FLOAT(53),CH CHAR(50) VAR,SF BIN FLOAT;                   00001000
      IF IED=0 THEN DO; SF=BF; CH=SF; END; ELSE CH=BF;                  00001100
      CALL DELETE(CH,' ');                                              00001200
      IF IED=1 THEN CALL REPLACE(CH,'E','D','0'B);                      00001300
      RETURN(CH);                                                       00001400
 END BFTC;                                                              00001500
```

```
* PROCESS;                                                              00000100
 BLNCD: PROC(CS) RETURNS(BIN FIXED);                                    00000200
 /*************************************************************** *****  00000300
  * THE DIFFERENCE BETWEEN THE NUMBER OF RIGHT AND LEFT PARENTHESES   *   00000400
  * IN THE CHARACTER STRING CS IS RETURNED                            *   00000500
  ********************************************************************/  00000600
      DCL CS CHAR(*) VAR, C CHAR(1) VAR, (IL,IR) INIT(0), I;            00000700
      DO I=1 TO LENGTH(CS);                                             00000800
         C=SUBSTR(CS,I,1);                                              00000900
         IF C='(' THEN IL=IL+1;                                         00001000
         IF C=')' THEN IR=IR+1;                                         00001100
      END;                                                              00001200
      RETURN (IL-IR);                                                   00001300
 END BLNCD;                                                             00001400
```

```
* PROCESS;                                                                00000100
 BREAKB: PROC(CS,C,BS);                                                   00000200
 /*************************************************************************00000300
  * THE CHARACTER STRING TO THE RIGHT OF THE CHARACTER VARIABLE C IN *     00000400
  * THE CHARACTER STRING CS IS PLACED IN THE CHARACTER STRING BS     *     00000500
  * C || BS IS DELETED FROM CS                                      *     00000600
  * EXAMPLE:  CS = 'ABCDE'                                          *     00000700
  *           CALL BREAKB(CS,'C',BS)                                *     00000800
  * GENERATES CS = 'AB', BS = 'DE'                                  *     00000900
  *************************************************************************/00001000
      DCL CS CHAR(*) VAR, C CHAR(1), BS CHAR(*) VAR,IX;                   00001100
      BS='';                                                              00001200
      DO I=LENGTH(CS) TO 1 BY -1;                                         00001300
         IF SUBSTR(CS,I,1)¬=C THEN GO TO LA;                              00001400
         IF I+1<LENGTH(CS) THEN BS=SUBSTR(CS,I+1);                        00001500
         IF I>1 THEN CS=SUBSTR(CS,1,I-1); ELSE CS='';                     00001600
         GO TO LB;                                                        00001700
 LA:  END;                                                                00001800
 LB:  END BREAKB;                                                         00001900
```

```
* PROCESS;                                                                00000100
 BREAKF: PROC(CS,C,BS);                                                   00000200
 /*************************************************************************00000300
  * THE CHARACTER STRING TO THE LEFT OF THE CHARACTER VARIABLE C IN  *     00000400
  * THE CHARACTER STRING CS IS PLACED IN THE CHARACTER STRING BS     *     00000500
  *  BS || C IS DELETED FROM CS                                      *     00000600
  * EXAMPLE:  CS = 'ABCDE'                                          *     00000700
  *           CALL BREAKF(CS,'C',BS)                                *     00000800
  * GENERATES CS = 'DE', BS = 'AB'                                  *     00000900
  *************************************************************************/00001000
      DCL CS CHAR(*, VAR, C CHAR(1), BS CHAR(*) VAR,IX;                   00001100
      BS='';                                                              00001200
      IX=INDEX(CS,C)-1;                                                   00001300
      IF IX=-1 THEN RETURN;                                               00001400
      IF IX<1 THEN GO TO LA;                                              00001500
      BS=SUBSTR(CS,1,IX);                                                 00001600
 LA:  IF IX+2<=LENGTH(CS) THEN CS=SUBSTR(CS,IX+2); ELSE CS='';            00001700
 LB:  END BREAKF;                                                         00001800
```

```
* PROCESS;                                                                00000100
 COUNT: PROC(CS,C) RETURNS(BIN FIXED);                                   00000200
 /*************************************************************************00000300
  * THE NUMBER OF TIMES THE CHARACTER STRING C APPEARS IN THE         *   00000400
  * CHARACTER STRING CS IS RETURNED                                   *   00000500
  *************************************************************************/00000600
      DCL CS CHAR(*) VAR, C CHAR(*) VAR,(MRKR,IC) INIT(0);                00000700
 LA:  IXC= INDEX(SUBSTR(CS,MRKR+1),C);                                    00000800
      MRKR=MRKR+IXC;                                                      00000900
      IF  IXC¬=0 THEN IC=IC+1;                                            00001000
      IF  IXC=0 | MRKR=LENGTH(CS) THEN RETURN(IC);                        00001100
      GO TO LA;                                                           00001200
 END COUNT;                                                               00001300
```

```
* PROCESS;                                                                00000100
 DELETE: PROC(CS,C);                                                     00000200
 /*************************************************************************00000300
  * EXAMPLE:  CS = 'ABCDE'              CS = 'A''BCD''E'              *   00000400
  *           CALL DELETE(CS,'C')       CALL DELETE(CS,'''')          *   00000500
  * GENERATES CS = 'ABDE'               CS = 'AE'                     *   00000600
  *************************************************************************/00000700
     DCL C CHAR(1), CS CHAR(*) VAR, I INIT(2),                            00000800
         FLIP BIN FIXED INIT(1),SYM CHAR(1) INIT('''');                   00000900
      CS=' ' || CS || ' ';                                                00001000
      IF C=SYM THEN GO TO LH;                                             00001100
      DO WHILE(I<LENGTH(CS));                                             00001200
         IF SUBSTR(CS,I,1)=C                                              00001300
          THEN CS=SUBSTR(CS,1,I-1) || SUBSTR(CS,I+1);                     00001400
          ELSE I=I+1;                                                     00001500
      END;                                                                00001600
      GO TO LIA;                                                          00001700
 LH:  DO WHILE(I<LENGTH(CS));                                             00001800
         IF SUBSTR(CS,I,1)=SYM                                            00001900
          THEN DO; FLIP=-FLIP; GO TO LHA; END;                            00002000
         IF FLIP<0 THEN                                                   00002100
 LHA:     DO; CS=SUBSTR(CS,1,I-1) || SUBSTR(CS,I+1); GO TO LI; END;       00002200
          ELSE I=I+1;                                                     00002300
 LI:  END;                                                                00002400
 LIA: CS=SUBSTR(CS,2,LENGTH(CS)-2);                                       00002500
 END DELETE;                                                              00002600
```

```
* PROCESS;                                                                    00000100
 LIBF: PROC(CS,IED);                                                         00000200
 /*****************************************************************        00000300
  * IF IED = 0 ALL DOUBLE PRECISION FORTRAN LIBRARY FUNCTIONS IN THE *        00000400
  * CHARACTER STRING CS ARE REPLACED WITH SINGLE PRECISION FUNCTIONS *        00000500
  * AND VICE VERSA IF IED = 1                                       *        00000600
  *****************************************************************/       00000700
      DCL IED,     CS CHAR(*) VAR,NF INIT(25),                              00000800
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1));           00000900
      DCL FN(25,2) CHAR(6) VAR INIT(                                        00001000
          'EXP',   'DEXP', 'ALOG10', 'DLOG10',  'ARSIN', 'DARSIN',         00001100
        'ARCOS', 'DARCOS',   'ATAN',  'DATAN',  'ATAN2', 'DATAN2',         00001200
          'SIN',   'DSIN',    'COS',   'DCOS',    'TAN',   'DTAN',         00001300
        'COTAN', 'DCOTAN',   'SQRT',  'DSQRT',   'TANH',  'DTANH',         00001400
         'SINH',  'DSINH',   'COSH',  'DCOSH',    'ERF',   'DERF',         00001500
         'ERFC',  'DERFC',  'GAMMA', 'DGAMMA', 'ALGAMA', 'DLGAMA',         00001600
         'AMOD',   'DMOD',    'ABS',   'DABS',  'AMAX1',  'DMAX1',         00001700
        'AMIN1',  'DMIN1',  'FLOAT', 'DFLOAT',   'SIGN',  'DSIGN',         00001800
         'ALOG',   'DLOG'  );                                               00001900
      DO I=1 TO NF;                                                         00002000
         CALL REPLACE(CS,FN(I,2-IED),FN(I,1+IED),'1'B);                     00002100
      END;                                                                  00002200
 END LIBF;                                                                  00002300
```

```
* PROCESS;                                                                00000100
 REPLACE: PROC(CS,SA,SB,OP);                                             00000200
 /*********************************************************************** 00000300
  * ALL APPEARANCES OF THE STRING SA IN THE CHARACTER STRING CS ARE  *    00000400
  * REPLACED WITH THE STRING SB IF THE BIT(1) VARIABLE OP = '0'B     *    00000500
  * IF OP = '1'B ONLY THOSE OCCURENCES OF SA BOUNDED ON THE RIGHT BY *    00000600
  * THE NULL CHARACTER OR +-/*)( ARE REPLACED WITH SB                *    00000700
  * EXAMPLE:     CS = 'A+BA'                                         *    00000800
  *              CALL REPLACE(CS,'A','Z','0'B)                       *    00000900
  * GENERATES    CS = 'Z+BZ'                                         *    00001000
  * WHILE        CALL REPLACE(CS,'A','Z','1'B)                       *    00001100
  * GENERATES    CS = 'Z+BA'                                         *    00001200
  * RESTRICTION: THE CHARACTER % MAY NOT APPEAR IN CS, SA, OR SB     *    00001300
  *********************************************************************** 00001400
      DCL                                                                 00001500
       CHECK ENTRY(BIN FIXED) RETURNS(BIT(1)),                            00001600
       VERIFY ENTRY(CHAR(*) VAR,CHAR(*) VAR) RETURNS(BIN FIXED),          00001700
       (CS,SA,SB) CHAR(*) VAR,OP BIT(1),OR BIT(') INIT('0'B);             00001800
 /*                                                                      */ 00001900
      LSA=LENGTH(SA);                                                    00002000
 LA:  MRKSA=INDEX(CS,SA);                                                00002100
      IF MRKSA=0 THEN GO TO LB;                                          00002200
      IF ¬OR THEN DO; CS=' '||CS||' '; MRKSA=MRKSA+1; END;               00002300
      OR='1'B;                                                           00002400
      CS=SUBSTR(CS,1,MRKSA-1) || '%' || SUBSTR(CS,MRKSA+LSA);            00002500
      GO TO LA;                                                          00002600
 LB:  IF ¬OR THEN RETURN;                                                00002700
      MRKSA=INDEX(CS,'%');                                               00002800
      IF MRKSA=0 THEN GO TO LC;                                          00002900
      IF ¬OP | CHECK(MRKSA)                                              00003000
       THEN CS=SUBSTR(CS,1,MRKSA-1) || SB || SUBSTR(CS,MRKSA+1);         00003100
       ELSE CS=SUBSTR(CS,1,MRKSA-1) || SA || SUBSTR(CS,MRKSA+1);         00003200
      GO TO LB;                                                          00003300
 LC:  CS=SUBSTR(CS,2,LENGTH(CS)-2);                                      00003400
 /*                                                                      */ 00003500
 CHECK: PROC(IX) RETURNS(BIT(1));                                        00003600
      DCL IX BIN FIXED,(IL,IR) BIN FIXED INT INIT(0);                    00003700
      IF IX-1>0 THEN IL=VERIFY(SUBSTR(CS,IX-1,1),'=(+-*/)');             00003800
      IF IX+1<=LENGTH(CS)                                                00003900
       THEN IR=VERIFY(SUBSTR(CS,IX+1,1),'=(+-*/)');                      00004000
      RETURN(IL+IR=0);                                                   00004100
 END CHECK;                                                              00004200
 END REPLACE;                                                            00004300
```

```
* PROCESS;                                                              00000100
 SPAN: PROC(CS,SA,SB,SC);                                              00000200
 /*****************************************************************   00000300
  * THE CHARACTER STRING IN CS SPANNED BY SA AND SB IS RETURNED IN SC*  00000400
  * EXAMPLE:   CS = 'ABCDEF'                                       *   00000500
  *            CALL SPAN(CS,'AB','F',SC)                           *   00000600
  * GENERATES  SC = 'CDE'                                          *   00000700
  *****************************************************************/  00000800
      DCL (CS,SA,SB,SC) CHAR(*) VAR;                                   00000900
      SC='';                                                           00001000
      ISTART=INDEX(CS,SA)+LENGTH(SA);                                  00001100
      ISTOP=INDEX(SUBSTR(CS,ISTART),SB);                               00001200
      IF ISTART>0&ISTOP>1 THEN SC=SUBSTR(CS,ISTART,ISTOP-1);           00001300
 END SPAN;                                                             00001400
```

```
* PROCESS;                                                              00000100
 TRIM: PROC(STRING);                                                   00000200
 /* ALL TRAILING BLANKS ARE DELETED FROM THE STRING CS */              00000300
          DCL STRING CHAR(*) VAR;                                      00000400
          DO I=LENGTH(STRING) TO 1 BY -1 WHILE (SUBSTR(STRING,I,1)=' ');00000500
          END;                                                         00000600
          IF I=0 THEN STRING=''; ELSE STRING=SUBSTR(STRING,1,I);       00000700
 END TRIM;                                                             00000800
```

APPENDIX B

```
      SUBROUTINE TAYLOR (TI,YI,TF,YF,NCF,NEQ,INIT,EPS,RANGE)          00000100
      IMPLICIT REAL*8 (A-H,O-Z)                                       00000200
      COMMON /COUNT/ KEV,KIS,IER                                      00000300
      COMMON /PAR/ HMIN                                               00000400
      REAL*8 YI(NCF,1),YF(NCF,1)                                      00000500
      LOGICAL*4 INIT                                                  00000600
      IF (.NOT.INIT)  GO TO 11                                        00000700
C**** INITIALIZATION                                                  00000800
      IER = 0                                                         00000900
      KEV = 0                                                         00001000
      KIS = 0                                                         00001100
      NCL = NCF - 1                                                   00001200
      EX = 1.D0/NCL                                                   00001300
      NEQ1 = NEQ + 1                                                  00001400
      CR = DSIGN(1.D0,RANGE)                                          00001500
      CALL INITAL(TI,YF,NCF)                                          00001600
      CALL COEFF (YF,NCF)                                             00001700
      DO 18  K = 1,NEQ                                                00001800
      DO 18  J = 1,NCF                                                00001900
   18 YI(J,K) = YF(J,K)                                               00002000
C**** COMPUTE R(H)                                                    00002100
   11 R = 0.D0                                                        00002200
      DO 10  K = 1,NEQ                                                00002300
      RJ = YI(NCF,K)                                                  00002400
      IF (YI(1,K).NE.0.D0)  RJ = RJ/YI(1,K)                           00002500
   10 R = DMAX1(R,DABS(RJ))                                           00002600
C**** COMPUTE H                                                       00002700
      IF (R.NE.0.D0)  GO TO 23                                        00002800
      H = RANGE                                                       00002900
      GO TO 24                                                        00003000
   23 H = CR*(EPS/R)**EX                                              00003100
   24 IF (DABS(H).GT.HMIN)  GO TO 17                                  00003200
C**** ERROR: H TOO SMALL                                              00003300
      IER = - 2                                                       00003400
      RETURN                                                          00003500
```

```
C**** TAKE A STEP                                                    00003600
   17 DO 12  K = 1,NEQ                                               00003700
   12 YF(1,K) = YI(NCL,K)                                            00003800
      DO 13  JJ = 2,NCL                                              00003900
      J = NCF - JJ                                                   00004000
      DO 13  K = 1,NEQ                                               00004100
   13 YF(1,K) = YI(J,K) + H*YF(1,K)                                  00004200
      TF = TI + H                                                    00004300
      YF(1,NEQ1) = TF                                                00004400
      CALL COEFF(YF,NCF)                                             00004500
      KEV = KEV + 1                                                  00004600
      RANGE = RANGE - H                                              00004700
      KIS = KIS + 1                                                  00004800
      RETURN                                                         00004900
      END                                                            00005000



      SUBROUTINE INTERP (TI,YI,NCF,NEQ,TW,W)                         00005100
      IMPLICIT REAL*8 (A-H,O-Z)                                      00005200
      REAL*8 YI(NCF,1),W(1)                                          00005300
      H = TW - TI                                                    00005400
      NCL = NCF - 1                                                  00005500
      DO 21  K = 1,NEQ                                               00005600
   21 W(K) = YI(NCL,K)                                               00005700
      DO 22  JJ = 2,NCL                                              00005800
      J = NCF - JJ                                                   00005900
      DO 22  K = 1,NEQ                                               00006000
   22 W(K) = YI(J,K) + H*W(K)                                        00006100
      RETURN                                                         00006200
      END                                                            00006300
```

```
      SUBROUTINE ZERO(T,F,A,TZ,B,C,N)                                   00000100
      IMPLICIT REAL*8(A-H,O-Z)                                          00000200
      DIMENSION A(N),B(N),C(N,N)                                        00000300
      C(1,1)=A(1)                                                       00000400
      YP=-F                                                             00000500
      B(1)=1.D0/C(1,1)                                                  00000600
      TZ=T+B(1)*YP                                                      00000700
      DO 500 K=2,N                                                      00000800
      C(K,1)=A(K)                                                       00000900
      DO 300 J=2,K                                                      00001000
      C(K,J)=0.D0                                                       00001100
      IMAX=K-J+1                                                        00001200
      DO 300 I=1,IMAX                                                   00001300
  300 C(K,J)=C(K,J)+C(K-I,J-1)*A(I)                                     00001400
      B(K)=0.0D0                                                        00001500
      IMAX=K-1                                                          00001600
      DO 400 I=1,IMAX                                                   00001700
  400 B(K)=B(K)+C(K,I)*B(I)                                             00001800
      B(K)=-B(K)/C(K,K)                                                 00001900
      YP=YP*(-F)                                                        00002000
      TZ=TZ+B(K)*YP                                                     00002100
  500 CONTINUE                                                          00002200
      RETURN                                                            00002300
      END                                                               00002400
```

APPENDIX C

In the following compilation of the recurrence coefficients for commonly used non-rational functions, it is assumed that the $k^{\underline{th}}$ Taylor coefficients for the function A(t) are known and the $k^{\underline{th}}$ coefficient ($k \geq 1$) for B=f(A) is sought.

$$A(t) = \sum_{j=o}^{\infty} a_j (t-t_o)^j$$

$$B(t) = \sum_{j=o}^{\infty} b_j (t-t_o)^j$$

For each function, the functional relationship is listed first, followed by the defining differential equation and finally by the Taylor coefficients. Many functions such as sin, cos, tan are derived from a coupled differential system and consequently are listed together. The symbol denotes differentiation with respect to the independent variable t.

$B = \exp(A)$

$B' = BA'$

$$b_k = \left(\sum_{j=1}^{k} j a_j b_{k-j}\right)/k$$

$B = \log_e(A)$, $a_o > 0$

$B' = A'/A$

$$b_1 = a_1/a_o$$

$$b_k = \left(a_k - \sum_{j=1}^{k-1} j a_{k-j} b_j \right)/k)/a_o \quad , \quad k \geq 2$$

$B = \log_{10}(A)$, $a_o > 0$

$B' = \alpha A'/A$, $\alpha = \log(e)$

$$b_1 = \alpha a_1/a_o$$

$$b_k = \left(\alpha a_k - \sum_{j=1}^{k-1} j a_{k-j} b_j \right)/k)/a_o \quad , \quad k \geq 2$$

$B = A^{\alpha}$, α real, $a_c > 0$

$B' = \alpha BA'/A$

$$b_k = \sum_{j=0}^{k-1} (\alpha - j(\alpha + 1)/k) b_j a_{k-j})/a_o$$

$B = \sin(A)$, $C = \cos(A)$, $D = \tan(A)$, $c_o \neq 0$

$B' = CA'$, $C' = -BA'$, $D = B/C$

$$b_k = \left(\sum_{j=1}^{k} j a_j c_{k-j}\right)/k$$

$$c_k = -\left(\sum_{j=1}^{k} j a_j b_{k-j}\right)/k$$

$$d_k = \left(b_k - \sum_{j=1}^{k} c_j d_{k-j}\right)/c_o$$

$B = \sinh(A)$, $C = \cosh(A)$, $D = \tanh(A)$, $c_o \neq 0$

$B' = CA'$, $C' = BA'$, $D = B/C$

$$b_k = \left(\sum_{j=1}^{k} j a_j c_{k-j}\right)/k$$

$$c_k = \left(\sum_{j=1}^{k} j a_j b_{k-j}\right)/k$$

$$d_k = \left(b_k - \sum_{j=1}^{k} c_j d_{k-j}\right)/c_o$$

The Taylor coefficients for the operations +, -, *, / are also included.

$C = A + B$

$$c_k = a_k + b_k$$

$C = A - B$

$$c_k = a_k - b_k$$

$C = A*B$

$$c_k = \sum_{j=o}^{k} a_{k-j} b_j$$

$C = A/B$

$$c_k = \left(a_k - \sum_{j=1}^{k} b_j c_{k-j}\right)/b_o$$

APPENDIX D

```
RMAT ENTRY
 LEVEL  K  TYPE  (C, O, V | E)                    R OP      A(1)          A(2)      A(3)
                 #Y(1,2)=1.0$
   1    2    0   Y(1,2)
   1    3    5   =
   1    4   -1   #2
   1    5    9   $
                 #Y(1,1)=Y(1)**2+3.D0*T**2$      1 $       Y(2)          #2        Y(2)
   1    2    0   Y(1,1)
   1    3    5   =
   1    4    0   Y(1)
   1    5   10   **
   1    6   -1   #3
   1    7    1   +
                                                 2 *       Y(1)          Y(1)      ?2
   1    8   -1   #4
   1    9    3   *
   1   10    0   T
   1   11   10   **
   1   12   -1   #3
   1   13    9   $                               3 *       T             T         ?3
                                                 4 *       #4            ?3        ?4
                                                 5 +       ?2            ?4        ?5
                                                 6 $       Y(1)          ?5        Y(1)
                                                 RECURRENCE MATRIX
                                                 1 $       Y(2)          #2        T
                                                 2 *       Y(1)          Y(1)      ?2
                                                 3 *       T             T         ?3
                                                 4 *       #4            ?3        ?4
                                                 5 +       ?2            ?4        ?5
                                                 6 $       Y(1)          ?5        Y(1)
                                                 OPTIMIZED RECURRENCE MATRIX
                                                 1 $       Y(2)          #2        T
                                                 2 *       Y(1)          Y(1)      ?2
                                                 3 *       T             T         ?3
                                                 4 *       #4            ?3        ?4
                                                 5 +       ?2            ?4        ?5
                                                 6 $       Y(1)          ?5        Y(1)
```

```
CONSTANT TABLE
# 1=0.5D0                # 2=1.0D0          # 3=2                # 4=3.D0
DMAT ENTRY  1, 2, 3, 4, 5, 6
D MATRIX
 1, 1 -1   1, 2 -1   1, 3 -1   1, 4 -1   1, 5 -1   1, 6 -1   2, 1 -1   2, 2 -1
 2, 3 -1   2, 4 -1   2, 5 -1   2, 6  0   3, 1 -1   3, 2 -1   3, 3 -1   3, 4 -1
 3, 5 -1   3, 6 -1   4, 1 -1   4, 2 -1   4, 3  1   4, 4 -1   4, 5 -1   4, 6 -1
 5, 1 -1   5, 2  1   5, 3  1   5, 4  1   5, 5 -1   5, 6  0   6, 1 -1   6, 2  1
 6, 3  1   6, 4  1   6, 5  1   6, 6  0

CORRESPONDENCE BETWEEN RECURRENCE MATRIX ROWS AND THE Y ARRAY
  1 2        2 3        3 4        4 5        5 6        6 1
```


N. SA-GSFC